MODBASE: A SciDB-Powered System for Large-Scale Distributed Storage and Analysis of MODIS Earth Remote Sensing Data

by

Gary Lee Planthaber, Jr.

Submitted to the Department of Electrical Engineering and Computer Science on

May 21, 2012

in Partial Fulfillment of the Requirements for the

Degree of Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology.

Author: _____

Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
May 21, 2012

Certified by: _____

Prof. Michael R. Stonebraker
Adjunct Professor, Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Thesis Supervisor
May 21, 2012

Certified by: _____

Prof. James Frew
Associate Professor, Bren School of Environmental Science and Management
University of California Santa Barbara
Thesis Co-Supervisor
May 21, 2012

Accepted by: _____

Prof. Dennis M. Freeman
Chairman, Master of Engineering Thesis Committee

MODBASE: A SciDB-Powered System for Large-Scale Distributed Storage and Analysis of MODIS Earth Remote Sensing Data

by

Gary Lee Planthaber, Jr.

Submitted to the Department of Electrical Engineering and Computer Science on

May 21, 2012

in Partial Fulfillment of the Requirements for the

Degree of Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology.

ABSTRACT

MODBASE, a collection of tools and practices built around the open source SciDB multidimensional data management and analytics software system, provides the Earth Science community with a powerful foundation for direct, ad-hoc analysis of large volumes of Level-1B data produced by the NASA Moderate Resolution Imaging Spectroradiometer (MODIS) instrument. This paper details the reasons for building the MODBASE system, its design and implementation, and puts it to the test on a series of practical Earth Science benchmarks using standard MODIS data granules.

Thesis Supervisor: Prof. Michael R. Stonebraker
Adjunct Professor
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology

Thesis Co-Supervisor: Prof. James Frew
Associate Professor
Bren School of Environmental Science and Management
University of California Santa Barbara

Acknowledgements

I want to thank my research advisors Prof. Mike Stonebraker and Prof. James Frew for all the support, wisdom and patience they provided throughout the course of this project. Both were very considerate and made certain I received all the resources I needed along the way. This project would not have gone as smoothly as it did without their guidance. Beyond this project, both have been amazing mentors who genuinely care not only about the research at hand, but also about my long-term success.

Given the multi-disciplinary nature of this research, it was necessary to acquire a basic understanding of topic areas beyond my core areas of expertise. Prof. Frew and Peter Slaughter of the Bren School of Environmental Science and Management at UCSB were great resources and very accommodating on my weeklong visit to California for my "Earth Science cram session". It really was after that visit that this project received a strong injection of adrenaline and started navigating quickly down the path becoming a reality. I also want to thank James Kuyper, Level 1 Lead of the MODIS Science Data Support Team of the Sigma Space Corporation. James exhibited great patience and willingness to give very detailed answers to some tough questions about the MODIS data, particularly around the topic of geolocation. Without his help, I could have lost weeks of time trying to overcome challenges presented by upsampling the MODIS geolocation product.

This project depended heavily on the SciDB database, so I naturally had a great need for support from the SciDB team. My need was met with unmatched courtesy, energetic enthusiasm and free food! I want to thank Marilyn Matz, Alex Poliakov, Paul Brown and the entire SciDB engineering team for their dedication to this project. I spent countless hours at their offices in Waltham, MA learning the inner workings of SciDB so that I could ultimately make it perform Earth Science benchmarks efficiently. I want to give a special thanks to Alex for working all day on his normal duties and then working another shift late into the evenings to support my questions. I can honestly say that, without the assistance I received from Alex and the rest of the SciDB team, this project would have not gotten off the ground in any meaningful way.

I want to thank Anne Hunter, for being the greatest administrator any department could ever hope to have. When I think back over my time spent in the MIT community, Anne stands out as a bright fixture in those memories. It so rare to find someone as dedicated and committed to their job and who goes so far above and beyond the call of duty on a daily basis. Anne truly paved the way for me to get to the point where this thesis could become a reality.

Finally, I want to thank my family and my loved ones for being so supportive of me and never failing to be there when I needed them. Without all of them, nothing of this would be meaningful or possible.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction and Background

1.1 Introduction

Many scientific, governmental, and commercial organizations can now cost-effectively collect, generate, and store data at rates that test the organization's ability to analyze the data using conventional storage and analytics tools. Relational database management systems (RDBMS), which have been a dominant technology for data storage and analysis for decades, have begun to age and are showing fundamental scaling limitations in the present era where large-scale horizontal scalability and distributed computation are required to perform complex analysis on very large sets of data [1].

While not every data storage and analysis application experiences the challenges described above, the number that do appears to be growing. In response, the market has coined the term "Big Data" to refer to both the classification of problems that cannot be easily addressed with conventional analysis platforms and to a new industry segment rapidly arising to fill the void. Though traditional RDBMS technology, with its maturity and support infrastructure, will likely remain the dominant technology for use in many conventional applications for the foreseeable future, it is evident that platforms built from the ground-up with a mandate to store and process very large volumes of data in a distributed manner will supplement or even displace RDBMS solutions for certain applications [2]. To meet the demands of some of the most challenging applications, such as the ones faced in many areas of science, database systems may well be in need of a complete, ground-up rewrite [3]. Because new systems constructed using techniques gleaned from decades of post RDBMS research often post a 1-2 order of magnitude

improvement in performance over conventional technologies in the market, it is likely they will also end up challenging expensive purpose-built systems created to achieve similar gains over legacy RDBMS systems.

With the landscape of database technology in a state of flux, a plethora of systems having widely varying specializations are competing in the overall marketplace [**4**]. Some examples of next-generation database technologies include: object-oriented, on-line analytical processing, column-oriented, parallel DBMSs, map-reduce, NoSQL, stream, and multi-dimensional among many others. Though many of the new technologies appear at first glance to compete directly with each other, in some cases apparent conflicts are actually better classified as complementary relationships [**5**], which can lead to confusion and misunderstanding.

As a natural result of the emergence of this complex new database technology landscape, it can be challenging for organizations, application engineers, and end-users to determine which technology may best suit their particular requirements. Apparently gone are the days when organizations can ensure a project's success simply by selecting the dominant RDBMS vendor. In such an environment, one of the most effective and powerful tools available to determine the applicability of a given technology to a given class of problem is a proof-of-concept (POC) project. POC projects serve to demonstrate the ability of a platform to perform particular tasks by designing, implementing, and ultimately characterizing the performance of a working prototype.

As will be detailed in the sections that follow, this thesis can be broadly characterized as a POC project aimed at characterizing the applicability of the SciDB multidimensional array analytics platform to perform tasks relevant to members of the Earth remote sensing community.

1.2 SciDB Primer

In 2007, the first extremely large databases conference (XLDB) convened, hosted by the Stanford Linear Accelerator Center (SLAC) [6], to discuss trends and needs unique to the handling of "extremely large databases", an ever-evolving term that presently refers to databases that may reach at least petabyte scale and which resist analysis by conventional tools and techniques. One precipitating factor to the convening of the first annual XLDB conference was the set of challenges faced by the Large Synoptic Survey Telescope (LSST) project team (part of SLAC) with storing and performing complex scientific analysis on the 50-100 petabytes of data expected to be collected by the project [7]. From the first 2007 XLDB conference onward, a community congealed around the core realization that better technologies must be developed to address the requirements of meaningfully analyzing such voluminous and complex datasets.

Following the XLDB 2007 conference was the generation of a set of common requirements [8] that served to represent some of the most salient features that next-generation analysis systems should implement. It was determined that the analysis requirements for science projects were not fundamentally different from many complex analysis problems being faced in industry and that a multidimensional array-oriented data model was specified despite the open recognition that it would not suit the needs of all areas of analysis. Additional requirements included specifications that the project would: be open-source, have a no-overwrite policy, have traceability (provenance), be based on a shared-nothing distributed architecture, have built-in extensibility, and support multiple language bindings. These requirements become some of the earliest objectives of the SciDB project. Around the same time frame, a standard science benchmark document [9] was released which outlined an analysis task loosely based on

workload pertinent to the field of astronomy. As is typically the case with any benchmark, the goal was to create a test against which the performance of multiple systems could be compared. The benchmark document further contained the results of implementing the benchmark on MySQL and an early incarnation of SciDB. Perhaps unsurprisingly, for the given benchmark, SciDB vastly outperformed MySQL overall. In Table 1, we can see the results of SciDB versus MySQL in the 2009 benchmark. In nearly all cases, SciDB showed its potential by outperforming MySQL by orders of magnitude.

DBMS	Dataset	Loading/Cooking [min]				Query Runtimes [min]									
		Load	Obsv	Group	Total	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Total
MySQL	*small*	760	110	2	872	123	21	393	0.4	0.36	0.6	0.6	49	50	638
	normal	770	200	90	1060	54	44	161	50	32	51	52	395	395	1234
	(scaleup)	(1.0)	(1.8)	(45)	(1.2)	(0.4)	(2.1)	(0.4)	(125)	(89)	(85)	(87)	(8.1)	(7.9)	(1.93)
SciDB	*small*	34	1.6	0.6	36	8.2	0.2	3.7	0.007	0.01	0.01	0.01	1.8	1.9	16
	normal	67	1.9	15	84	3.6	0.07	1.7	0.015	0.017	0.02	0.11	2.2	2.3	10
	(scaleup)	(2.0)	(1.2)	(25)	(2.3)	(0.4)	(0.4)	(0.4)	(2.1)	(1.7)	(2)	(11)	(1.2)	(1.2)	(0.63)
(MySQL	*small*	(22)	(69)	(3.3)	(24)	(15)	(105)	(106)	(57)	(36)	(60)	(60)	(27)	(26)	(40)
/SciDB)	*normal*	(12)	(105)	(6)	(13)	(15)	(630)	(95)	(3330)	(1880)	(2550)	(470)	(180)	(170)	(120)

Table 1: 2009 Results of MySQL versus SciDB [9]

In 2009 Paradigm4, Inc. was incorporated to provide a commercial version of SciDB and SciDB, Inc. was incorporated as a non-profit organization to provide the community version. With SciDB, as with many open source products, there is a dual model providing a usable, freely available community version of the product as well as a generally more feature rich commercial version with professional support options. Also in the same year, the SciDB system was officially announced to the academic world as part of the 2009 VLDB conference **[10]** and a specified feature set very closely matching that of the requirements presented earlier that year was indicated. From 2010 through 2011, as additional development work on SciDB took place, more concrete details emerged about the inner working of the database, the reasoning behind

particular design decisions [**11**], and the overall architecture [**12**]. As of April 2012, version 12.3 of SciDB has just been released and provides enough generalized functionality to perform significant analysis.

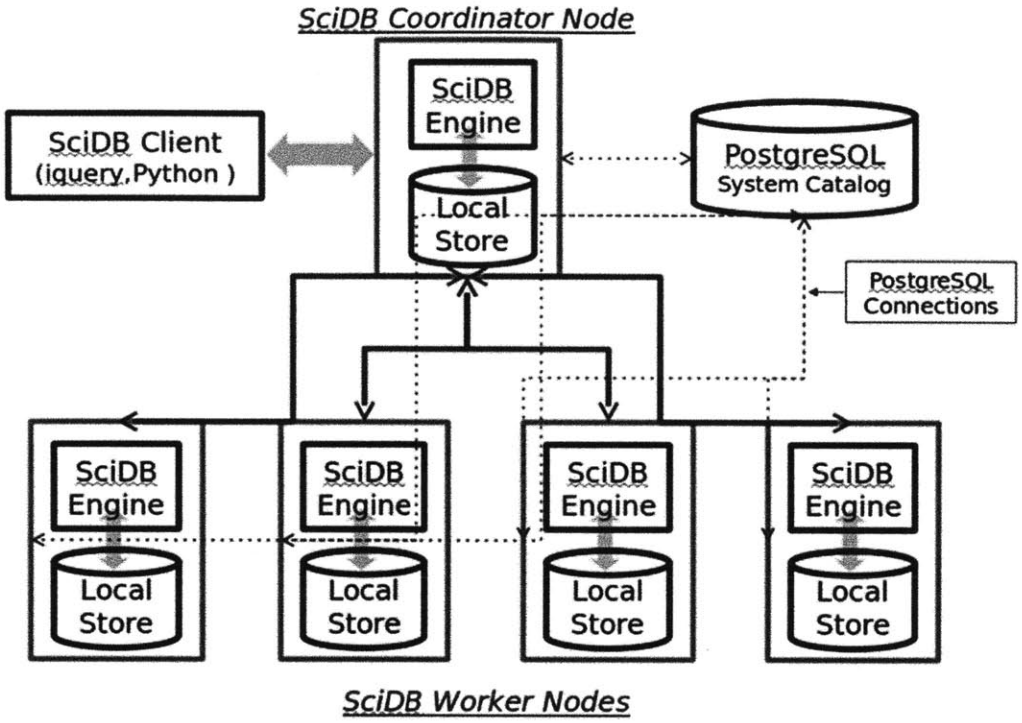

Figure 1: SciDB System Schematic [13]

While the core of SciDB is written primarily in C++, some supporting utilities (e.g., iquery) are written in Python. In terms of platforms, SciDB is currently supported on the Linux operating system, particularly the Red Hat and Ubuntu distributions. PostgreSQL is currently a required dependency of the SciDB system as it is leveraged for providing support for the metadata catalog. True to the early stated requirements, SciDB can run as a cluster of one or more shared-nothing instances on the same machine and/or across a network. Figure 1 shows the

system schematic for a 5-instance SciDB cluster. Clients interact with the database though a SciDB client application and the multiple database instances work together to automatically distribute storage and processing requirements with one instance acting as a coordinator.

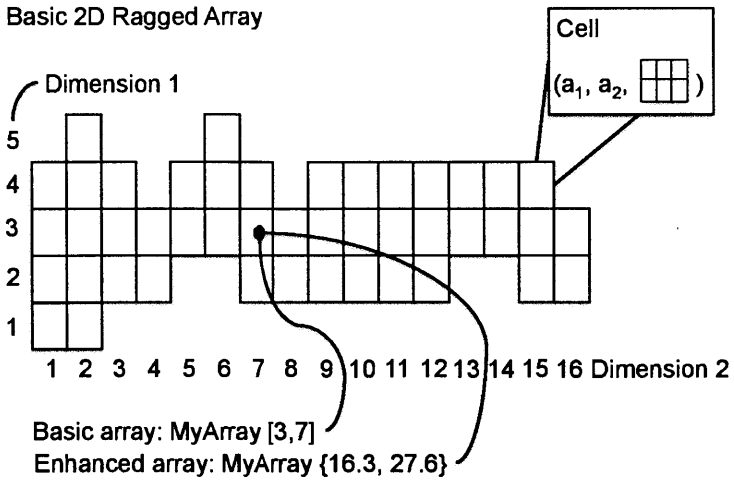

Figure 2: Basic 2D Ragged SciDB Array [10]

At the core of the SciDB data model is a multidimensional array. In Figure 2, we can see a representation of a ragged (sparse) 2 dimensional SciDB array. SciDB arrays are defined in terms of both dimensions and attributes [**13**]. Dimensions are always signed 64-bit integers. Other data types such as strings and floats can also be used as dimensions, but they are ultimately mapped to 64-bit signed integer values by the system. Attributes, on the other hand, can take on one of many data types and exist within the cells representing the intersections of the dimensional indices. An important realization about multidimensional arrays is that they may be either dense or sparse depending on what fraction of the cells in the array are empty. SciDB has built-in support allowing the efficient handling of sparse arrays, but the dimensionality and sparseness of an array may have an impact on performance.

In SciDB, one can think of an array as being composed of one or more pieces called chunks. Chunks are somewhat analogous to the concept of pages on disk, but are defined on a per-array basis. Chunks allow potentially large logical arrays to be physically partitioned into smaller, more manageable pieces. When defining an array in SciDB, one specifies not only the dimensions and their valid ranges, but also the number of cells along each dimension that should be included in a chunk. Furthermore, since SciDB is also a column-oriented database, each chunk contains the values for only a single attribute. In Figure 3, we see a demonstration of how the SciDB storage manager physically stores the values of a logical array. The logical array is first partitioned by attribute and then each attribute array is decomposed into physical chunks.

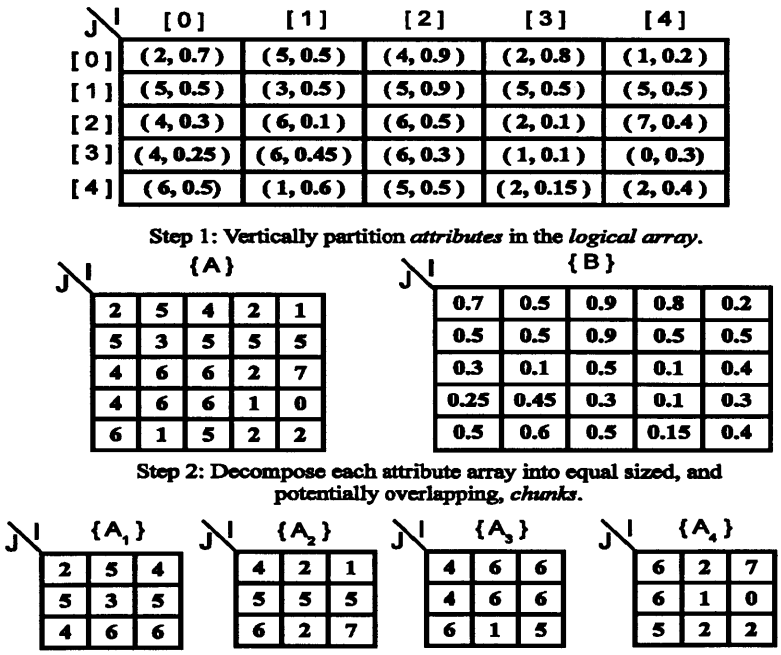

Figure 3: SciDB Storage Manager Behavior [11]

Presently, chunk sizing is a very important concept to architects of SciDB schemas and selecting optimal values for chunk sizes is critical to realizing the full performance potential of the SciDB database. Chunks are also allowed to overlap each other. In Figure 4, we see a visualization of 3 chunks of size 5x5, which have a specified overlap of 2 (darker gray). This feature can provide enhanced performance for certain operators, which may require accessing cells that are part of an adjacent chunk. If one chunk resides on one instance and the adjacent chunk on another instance, it might be necessary to transmit chunks over the network to perform an operation. With appropriate overlaps defined, this transfer may be avoided at the expense of storing extra information per chunk.

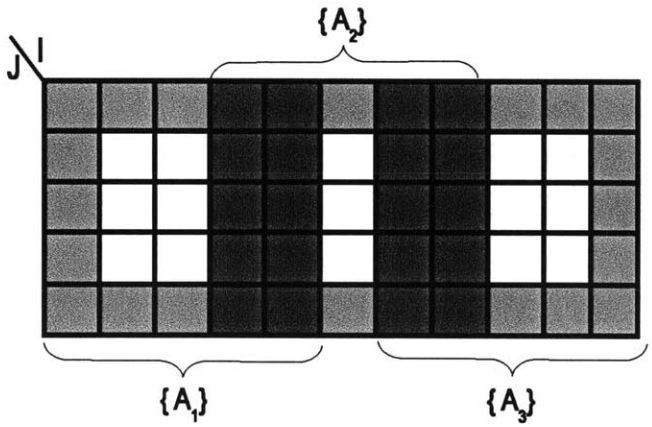

Figure 4: Visualization of Chunk Overlap [11]

Finally, interaction with SciDB is accomplished through the use of language bindings (such as Python) or via the iquery utility provided with the database. Two languages are available for executing commands against the database: an SQL-like language called Array Query Language (AQL) and a functional language called Array Functional Language (AFL).

1.3 MODIS Primer

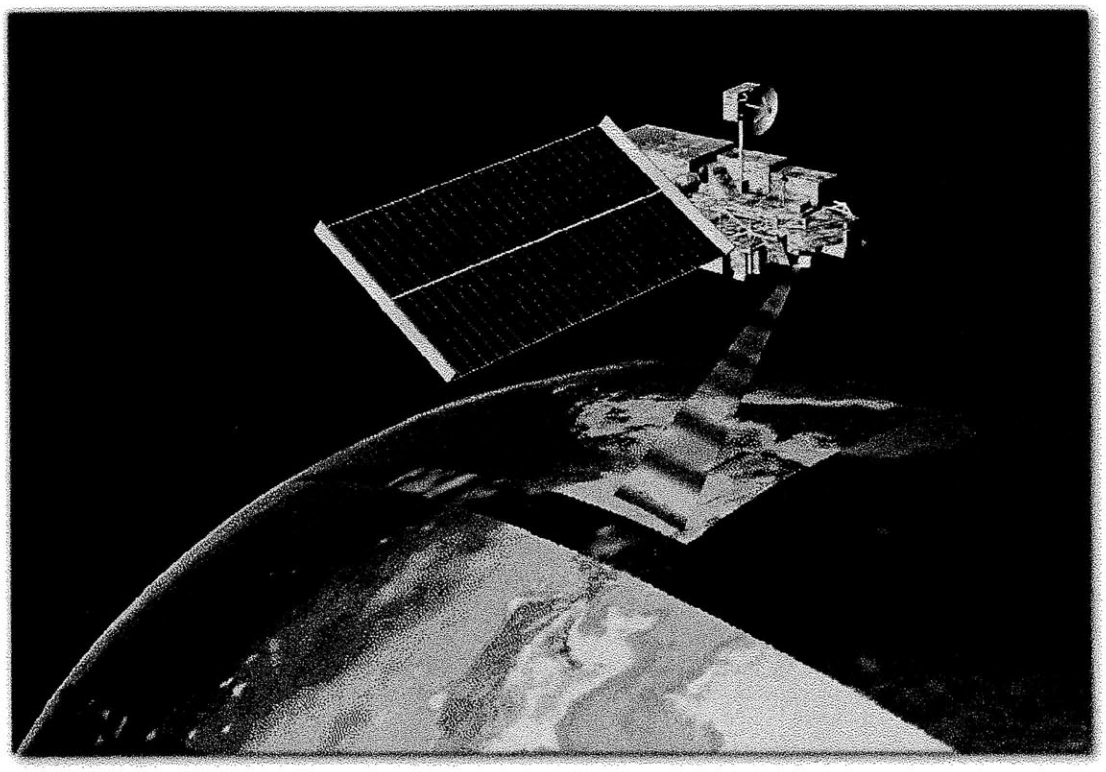

Figure 5: Illustration of MODIS Scanning Earth from the Terra Satellite [14]

The Moderate Resolution Imaging Spectroradiometer (MODIS) **[15]**, is a scientific instrument built for use by the National Aeronautics and Space Administration (NASA) as part of its Earth Observing System (EOS) program **[16]**. The MODIS instrument **[14]** has been installed on two separate NASA satellites: Terra (a.k.a. EOS AM, Launched December 1999) **[17]** and Aqua (a.k.a. EOS PM, Launched May 2002) **[18]**. Figure 5 illustrates the Terra satellite in the process of scanning a swath of the earth. MODIS is designed to collect detailed remote sensing data relevant to Earth scientists. MODIS senses data for 36 spectral bands and 3 spatial resolutions and continuously collects data at rates up to 11 Mbps. Refer to Figure 6 for detailed

specifications for the MODIS instrument. Aqua and Terra both have polar orbits that enable them to encircle the Earth approximately every 90 minutes and view nearly the entire surface of the planet every 1-2 days.

NASA not only provides raw MODIS data to the scientific community free of charge, but also offers a very rich, hierarchical ecosystem of free derived data products [**19**]. These data products are divided into the following categories: low-level calibration, atmosphere, land, cryosphere, and ocean products. Each category has several data products available beneath it and each data product is the result of established analytical techniques in the field of Earth Science. NASA had to make decisions and assumptions about which analytical techniques and final representations it should use for its ecosystem of products. Though these decisions are ostensibly made with the goal of meeting the needs of the greatest number of potential consumers of the products, no set of decisions could ever perfectly suit every research group that makes use of MODIS data and many Earth Scientists would like to have the flexibility to efficiently perform analysis starting from low-level data.

In the area of support, NASA and other organizations have contributed a variety of tools, which make the processing of data products even more manageable (e.g., [**20**]). These tools are used for data extraction, re-projection, gridding, stitching, sub-sampling among other things and most of them are also freely available and generally well documented. NASA presumably provides tools and packaged products because of the difficulty many research groups may experience with writing software and analyzing the large volumes of raw MODIS data themselves.

Orbit:	705 km, 10:30 a.m. descending node or 1:30 p.m. ascending node, sun-synchronous, near-polar, circular
Scan Rate:	20.3 rpm, cross track
Swath Dimensions:	2330 km (across track) by 10 km (along track at nadir)
Telescope:	17.78 cm diam. off-axis, afocal (collimated), with intermediate field stop
Size:	1.0 x 1.6 x 1.0 m
Weight:	250 kg
Power:	225 W (orbital average)
Data Rate:	11 Mbps (peak daytime)
Quantization:	12 bits
Spatial Resolution:	250 m (bands 1-2)
(at nadir):	500 m (bands 3-7), 1000 m (bands 8-36)
Design Life:	5 years

Primary Use	Band	Bandwidth[1]	Spectral Radiance[2]	Required SNR[3]
Land/Cloud	1	620-670	21.8	128
Boundaries	2	841-876	24.7	201
Land/Cloud	3	459-479	35.3	243
Properties	4	545-565	29.0	228
	5	1230-1250	5.4	74
	6	1628-1652	7.3	275
	7	2105-2155	1.0	110
Ocean color/	8	405-420	44.9	880
Phytoplankton/	9	438-448	41.9	838
Biogeochemistry	10	483-493	32.1	802
	11	526-536	27.9	754
	12	546-556	21.0	750
	13	662-672	9.5	910
	14	673-683	8.7	1087
	15	743-753	10.2	586
	16	862-877	6.2	516
Atmospheric	17	890-920	10.0	167
Water Vapor	18	931-941	3.6	57
	19	915-965	15.0	250

Primary Use	Band	Bandwidth[1]	Spectral Radiance[2]	Required NEΔT(K)[4]
Surface/Cloud	20	3.660-3.840	0.45	0.05
Temperature	21	3.929-3.989	2.38	2.00
	22	3.929-3.989	0.67	0.07
	23	4.020-4.080	0.79	0.07
Atmospheric	24	4.433-4.498	0.17	0.25
Temperature	25	4.482-4.549	0.59	0.25
Cirrus Clouds	26	1.360-1.390	6.00	150[3]
Water Vapor	27	6.535-6.895	1.16	0.25
	28	7.175-7.475	2.18	0.25
	29	8.400-8.700	9.58	0.05
Ozone	30	9.580-9.880	3.69	0.25
Surface/Cloud	31	10.780-11.280	9.55	0.05
Temperature	32	11.770-12.270	8.94	0.05
Cloud Top	33	13.185-13.485	4.52	0.25
Altitude	34	13.485-13.785	3.76	0.25
	35	13.785-14.085	3.11	0.25
	36	14.085-14.385	2.08	0.35

[1]Bands 1 to 19, nm; Bands 20-36, μm
[2](W/m^2-μm-sr)
[3]SNR=Signal-to-noise ratio
[4]NEΔT=Noise-equivalent temperature difference
} Performance goal is 30%-40% better than required

Figure 6: MODIS Technical Specifications [21]

MODIS products are provided using a highly condensed and optimized binary file format called Hierarchical Data Format (HDF) with EOS-specific extensions (a.k.a. HDF-EOS [22]). According to the HDF Group, the organization that oversees the HDF format standards, NASA will provide 15 petabytes of data in the HDF-EOS format during the 15-year duration of the EOS project [23]. Any tools that perform down-stream processing of MODIS data must first be able to read data from the HDF-EOS file format. To aid in performing that inevitable task, the HDF Group and others offer libraries and tools enabling a variety of programming languages to access the contents of HDF files [24].

In summary, MODIS is not only an instrument built to collect Earth Science data, but also an entire set of products, tools and support put in place to enable very efficient study of various aspects of Earth Science. For common research cases, the benefits of this integrated approach to the scientific community are innumerable. As one can imagine, however, researchers whose work falls outside the common lanes supported by NASA still face significant challenges.

1.4 Problem Statement

While many Earth scientists benefit from the significant standardized product offerings that are made available by NASA, there are also many occasions where research groups want to perform analysis on the raw or calibrated data before any additional analysis or projection is performed. An obvious example is for research that generates analytical results not directly covered by any NASA product, but several other examples are conceivable that would necessitate processing from raw or calibrated data.

Without being able to benefit directly from the purpose-built infrastructure provided by NASA in these cases, research groups must be equipped and prepared to store and process very large, low-level datasets packaged in HDF-EOS file format. The natural and typical first response is to construct purpose-built software systems that must be able to extract data from HDF-EOS, hard-code the desired analysis, manage intermediate byproducts, and ultimately store the desired results.

Performing those steps efficiently, however, becomes a potentially costly challenge when dealing with the large MODIS datasets. Experience with writing highly parallel, error-prone code for distributed computing, both across processor cores and across physical machines in a networked cluster, is a prerequisite for this class of large-scale data analysis. Not all Earth Science research groups possess the time and resources to design, execute, and validate complex software engineering projects. Additionally, any changes made to underlying code or assumptions during the course of the project may make it necessary to perform some or all of the resource-intensive processing steps over and over again.

What is needed is a system that is capable of importing large volumes of HDF-EOS data and automatically handles the issues of parallel storage and computation management, freeing researchers to focus on the core analysis tasks at hand while enabling high-performance, ad-hoc analysis. Such a system, constructed with sufficient capacity, could ingest most or all of the available MODIS data set and could be centralized in a high-performance computing center and shared amongst various research organizations wishing to perform various analyses based upon the same raw data.

1.5 Objective

The primary objective of this project is to design, implement, and characterize the performance of a system for analyzing MODIS data, built around and upon the SciDB multidimensional array database. Using SciDB as the core of the system is a natural choice because of the multidimensional nature of the MODIS data and the inherent horizontal scalability of the SciDB platform. The resulting system should be able to import MODIS data in HDF-EOS files, transparently persist the imported data across one or more database instances, support distributed analysis across the database instances, and enable ad-hoc analysis without requiring advanced programming skills on the part of the researcher. Detailed performance metrics should be collected for each stage of the processing pipeline as well as for a small set of simple Earth Science analysis benchmarks.

1.6 Related Work

Some early work that is relevant to this project includes the Sequoia 2000 project **[25]** and its corresponding benchmark **[26]**. Many of the same challenges and approaches that were taken then are still applicable now. Other relevant benchmarking work done using SciDB to perform analysis in other scientific domains includes the aforementioned "Standard Science DBMS Benchmark" **[9]** and "Using SciDB to Support Photon Science Data Analysis" **[27]**. Finally, this work is also directly influenced by use cases **[28]** written by representatives of various research areas including Optical and Radio Astronomy, Earth Remote Sensing, Environmental Observation and Modeling, Seismology, ARM Climate Research and others.

Chapter 2: MODBASE System Overview

2.1 Design Considerations

2.1.1 Pipelined Workflows

A primary goal of this project is to demonstrate a system that could ultimately be used by the Earth Science community to accomplish intensive analysis of large volumes of MODIS data. It is important to consider the potential end users and the way they are already accustomed to working. In general, it appears that many Earth Science projects are making use of pipelined workflows made up of independent, specialized tools and utilities that, when strung together, produce desired results. Since the nature of the analysis being performed seems a natural fit for the workflow metaphor, this project strives to use a similar structure.

2.1.2 Traceability Management

It is important to realize that the deeper and more complicated workflows get, the more inflexible and error prone they may become. Additional workflow stages necessitate the storage and management of more intermediate byproducts for the sake of debugging and provenance. Researchers who wish to have full traceability through their analysis must persist not only the source and result data, but all intermediate data as well. Hence, this system should ideally support the ability to manage the persistence of all intermediate data in as consistent a way as possible. There should also be a high-level bias to keep all workflows moving forward only such that a given stage in the workflow is dependent only on the results of earlier stages. In general, it is not anticipated that cyclic workflows should be required for most Earth Science analysis.

2.1.3 Minimization of Rework

Where possible, workflows should be designed in such a way to minimize the amount of costly rework that must be performed when an underlying analysis parameter or technique is changed. Accomplishing minimum rework often amounts to performing the most general, unavoidable, and uncontroversial static analysis steps early in the workflow and persisting those results so that later analysis can at worst fall back to those results when changes are made rather than having to start from the very beginning of the workflow. In some cases this principle may affect the ordering of workflow stages, but in other cases it may simply dictate that more complex workflow stages be broken up into sub stages, permitting the establishment of "checkpoints".

2.1.4 Selective Denormalization

As is the case for any system responsible for loading, persisting and analyzing data, important decisions have to be made for how best to represent data in such a way as to enable efficient analysis while also being mindful of storage and memory constraints. There are often tradeoffs between representations that minimize storage requirements and representations that maximize the performance of analysis. Though SciDB is not a relational database, it is not immune to the possibility of benefiting from selective denormalization.

2.1.5 Parallelization

A conscious effort should be made to recognize and exploit natural parallelizability of processing, storage and analysis. This consideration has become more salient as trends are moving toward massively distributed storage and computing systems. Failing to identify opportunities for parallelization can leave modern systems grossly underutilized.

2.2 High-Level Data Flow

Figure 7 shows a high-level view of how data moves through the MODBASE system. Data originates in HDF files provided by NASA and is first processed by a preprocessor that performs several functions that will be covered in detail in later sections. The output of the preprocessor is comma-separated text files (CSV), which are processed by another tool called that converts CSV into SciDB dense load format in a way that supports parallel loading of data into SciDB. Once data is in SciDB and has been analyzed, it can optionally be exported to an external utility (e.g., MATLAB or R) for visualization purposes.

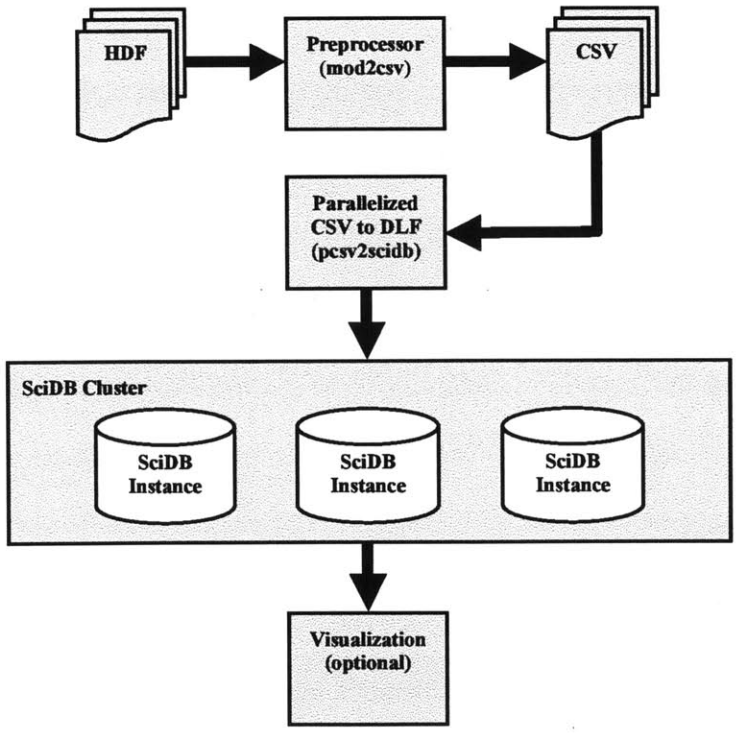

Figure 7: MODBASE High-Level Data Flow Diagram

2.3 System Components

2.3.1 Hardware

A single server costing on the order of $5000 was used for all benchmarking done in the course of this project. Detailed specifications for the server can be found in the appendices. The server's available memory and processor cores allowed for benchmarking SciDB clusters of between 1 and 5 instances running on the same physical machine.

2.3.2 Packaged Software

Aside from a suitable Linux OS installation, packaged software components used as part of the MODBASE system include: the SciDB database server, the iquery utility that comes packaged as part of the SciDB distribution, and a suitable version of the PostgreSQL RDBMS. MATLAB was also used for the purposes of displaying resulting image data.

2.3.3 Custom Software

Custom software components provide the "glue" needed to create a complete system for MODIS data analysis. Table 2 lists basic information about the custom software components of the MODBASE system. These components are covered in more detail later.

Name	Language	Platforms	Purpose
mod2csv	Java	Any Java	Preprocess MODIS data and convert from HDF to CSV.
pcsv2scidb	Java	Any Java	Convert CSV to parallelized SciDB DLF format.
load2scidb	Bash Script	Linux	Coordinates conversion and loading of CSV data.
rgb_composite	Bash Script	Linux	Benchmark for creating an RGB composite image.
ndvi	Bash Script	Linux	Benchmark for performing NDVI analysis.
ndsi_snowmap	Bash Script	Linux	Benchmark for performing NDSI+Snowmap analysis.
grid	Bash Script	Linux	Benchmark for gridding result data.
visualize	MATLAB	Any MATLAB	Creates image plots of analysis results.

Table 2: MODBASE Custom Software Components

Chapter 3: Preprocessing MODIS Data

3.1 Overview

This chapter details the design and operation of the preprocessor component (mod2csv) of the MODBASE system. At the highest level, the mod2csv utility ingests pairs of two different MODIS data products in HDF format and emits CSV files containing data useful for typical Earth Science analyses. Since it is stand-alone and emits CSV, the mod2csv utility can be used independently of SciDB and can be used with other tools and databases as well. This flexibility is especially helpful for providing a common starting point for benchmarking multiple systems against each other. One drawback of the decision to emit CSV is that the resulting text files are significantly larger than the highly compressed binary HDF files from which they are derived.

3.2 Inputs

The first input that must be specified to mod2csv utility is the path to one or more MODIS Level 1B Calibrated Radiances (MOD02) files. The mod2csv utility opens each specified MOD02 file and parses its metadata. In the process, mod2csv finds the name of the associated MODIS Level 1 Geolocation (MOD03) file ID in the metadata (ANCILLARYINPUTPOINTER) and searches for the MOD03 file in the same path as the MOD02 file. An error is thrown if either file cannot be opened. Thus, a precondition to using the mod2csv utility is that MOD02 files and their corresponding MOD03 files should be placed in the same directory prior to processing them. In addition to specifying the MOD02 input files, a user can also specify a desired output folder by adding a "-o <output folder>" to the command line options when invoking the mod2csv utility.

3.3 Geolocation Data Upsampling

Despite the fact that band measurement data is collected by MODIS at 3 different spatial resolutions (250m, 500m and 1km), NASA provides its MOD03 Geolocation product only at 1km resolution. Therefore, when preprocessing 250m and 500m data, it is necessary to upsample from the lower resolution 1km geolocation data in order to attribute geolocation values to each individual measurement. This geolocation upsampling is one of the major functions provided automatically by the mod2csv utility. All values are upsampled on a per-scan basis using bilinear interpolation and extrapolation, which has been shown to provide sufficient accuracy in this domain when compared to more computationally intensive interpolation techniques [**29**].

3.3.1 Upsampled Data Fields

Table 3 lists all data fields that are captured from the MOD03 data product and upsampled when applicable. All geolocation data fields are upsampled using bilinear interpolation and extrapolation. Some fields must be upsampled in a manner that honors the circular nature of their domains. Particularly, certain angular values have a discontinuity where their values change from 180 to -180 instantaneously. The mod2csv utility handles these special cases automatically during the upsampling process.

Field	Notes
/MODIS_Swath_Type_GEO/Geolocation Fields/Longitude	Circular Interpolation/Extrapolation Used
/MODIS_Swath_Type_GEO/Geolocation Fields/Latitude	Normal
/MODIS_Swath_Type_GEO/Data Fields/Height	Normal
/MODIS_Swath_Type_GEO/Data Fields/SensorZenith	Normal
/MODIS_Swath_Type_GEO/Data Fields/SensorAzimuth	Circular Interpolation/Extrapolation Used
/MODIS_Swath_Type_GEO/Data Fields/Range	Normal
/MODIS_Swath_Type_GEO/Data Fields/SolarZenith	Normal
/MODIS_Swath_Type_GEO/Data Fields/SolarAzimuth	Circular Interpolation/Extrapolation Used
/MODIS_Swath_Type_GEO/Data Fields/Land/SeaMask	Normal

Table 3: Upsampled Geolocation Data Fields

3.3.2 Upsampling Demonstration

The following examples demonstrate how upsampling is performed for the two cases where we want to start with the 1km data provided by NASA and end up with geolocation data corresponding to 500m and 250m measurements. As described in the Algorithm Theoretical Basis Document (ATBD) pertinent to MODIS Level 1A Earth Location [**30**], MODIS pixel values are determined by triangular weighting functions that relate values at each spatial resolution. These triangular weighting functions break the commonsense notion that (4) 500m pixels or (16) 250m pixels are equivalent to (1) 1km pixel. Additionally, the location of the center of each pixel is therefore also offset in a predictable way as shown in Figure 8.

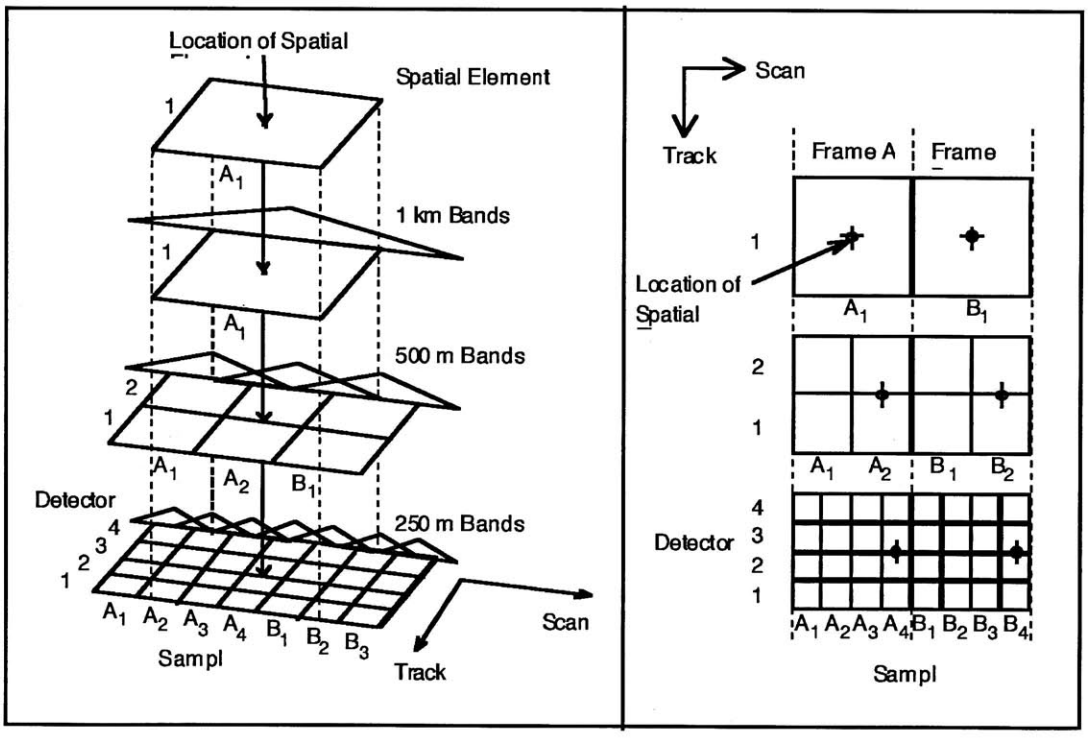

Figure 8: Relative Location of MODIS 1km Spatial Element by Resolution [30]

Bow Tie Effect

The bow tie effect, shown in Figure 9, is an artifact of the very wide scan field of the MODIS instrument. At high scan angles, the area covered by each pixel becomes larger and each scan bows out as we can see in Figure 10. As a result, scans may have significant overlap with prior ones and upsampling must be performed on a per-scan basis.

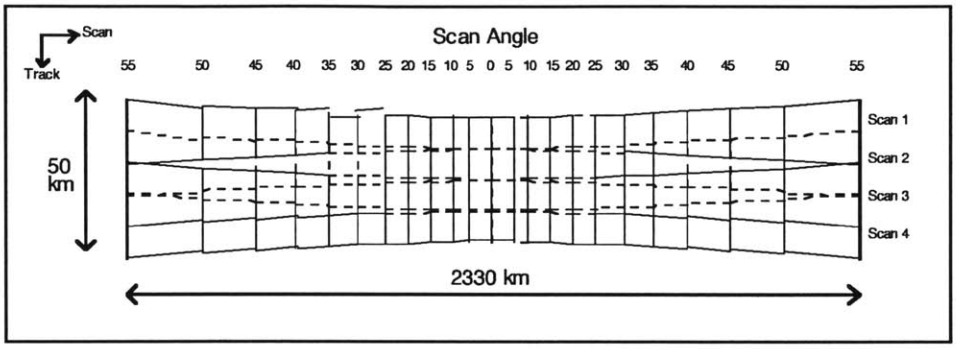

Figure 9: Panoramic Bow Tie Effect [30]

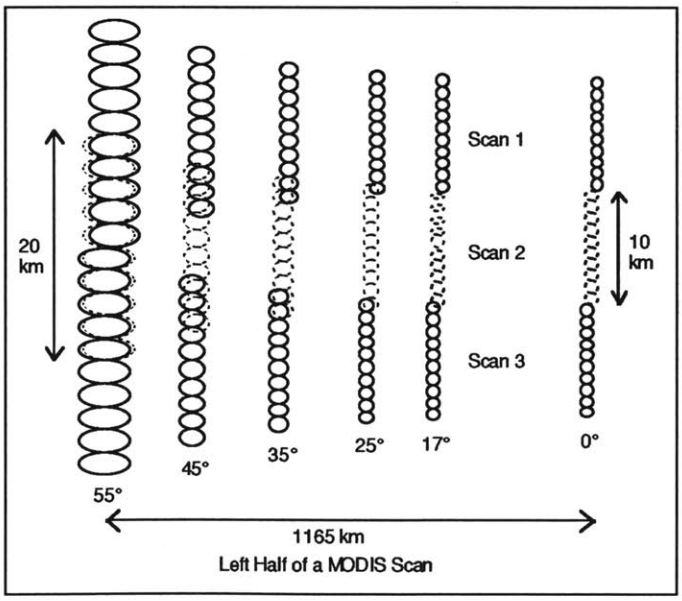

Figure 10: Pixel Size Growth and Overlap within a Scan [30]

Upsampling 1km to 500m

The relationship between 1km pixel centers and 500m pixel centers can be seen visually in Figure 11. In the track direction we see the expected 1:2 relationship, since the triangular weighting functions do not apply in the track direction. In the scan direction, however, we can see the result of the triangular weighting functions in the offset of 500m pixels versus their corresponding 1km pixels. One can also see what appears to be 1 missing 500m pixel at the start of the scans, which was purposefully left out by NASA in order to apparently preserve a 1:4 data ratio.

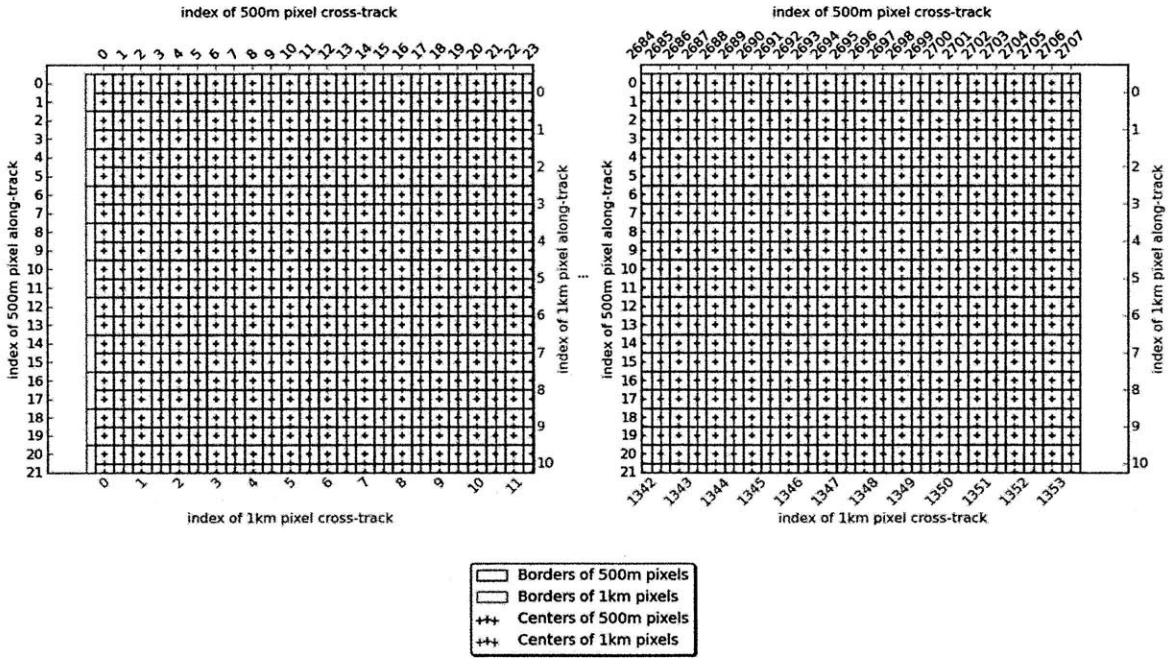

Figure 11: Relative Indices of 1km versus 500m Pixels [31]

1km and 500m Pixel Centers Near Nadir

Figure 12 shows a plot of a section of 1km pixel centers (plusses) relative to the 500m pixel centers (circles) that were calculated by mod2csv. This section of pixel centers is located around nadir (looking straight down on the planet from the satellite) and exhibit fairly regular spacing and no overlap, as expected.

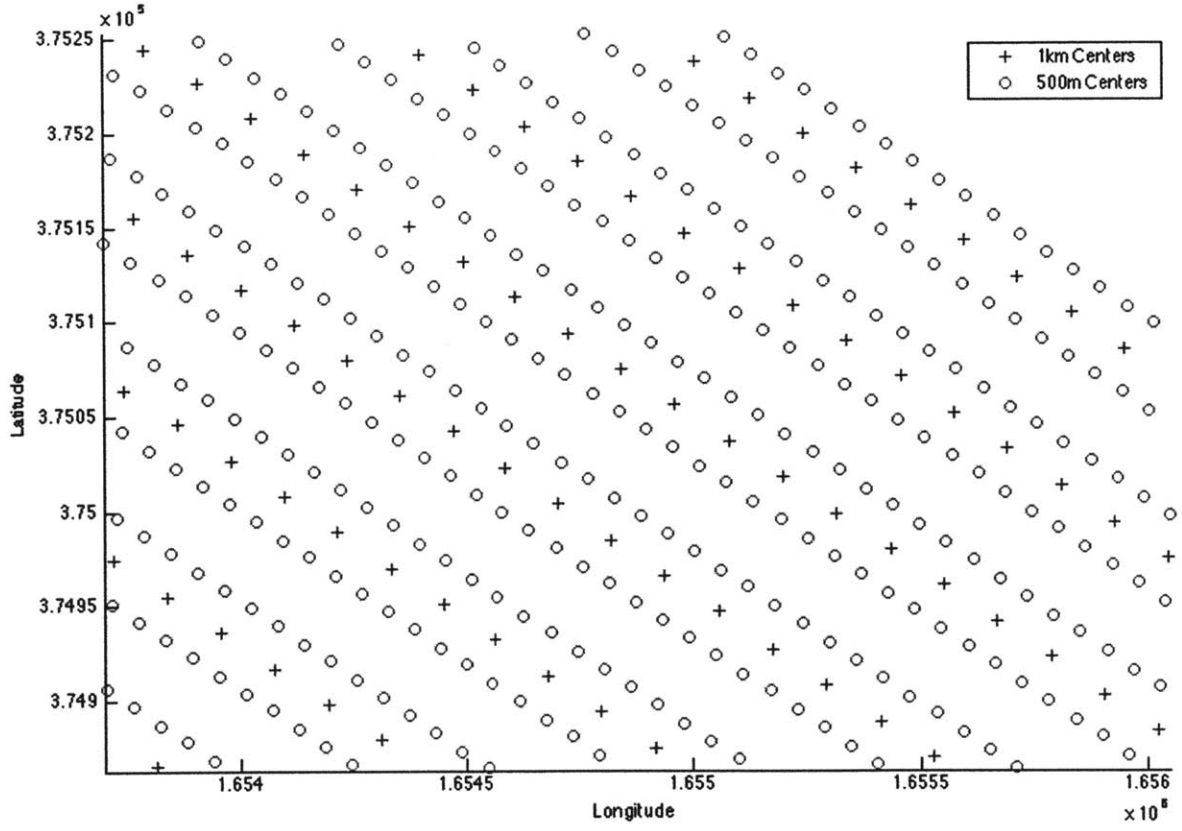

Figure 12: 1km and 500m Pixel Centers Near Nadir

1km and 500m Pixel Centers Near Start of Scan

Figure 13 shows a plot of a section of 1km pixel centers (plusses) relative to the 500m pixel centers (circles) that were calculated by mod2csv. In contrast to the prior figure, this section of pixel centers is located at the start of the 2 displayed scans and zoomed a bit further out so that the bow tie effect can be seen very clearly. It is noticeable that the centers of some pixels from the second scan fall in nearly the same geographic location as some of the pixel centers from the scan made before it.

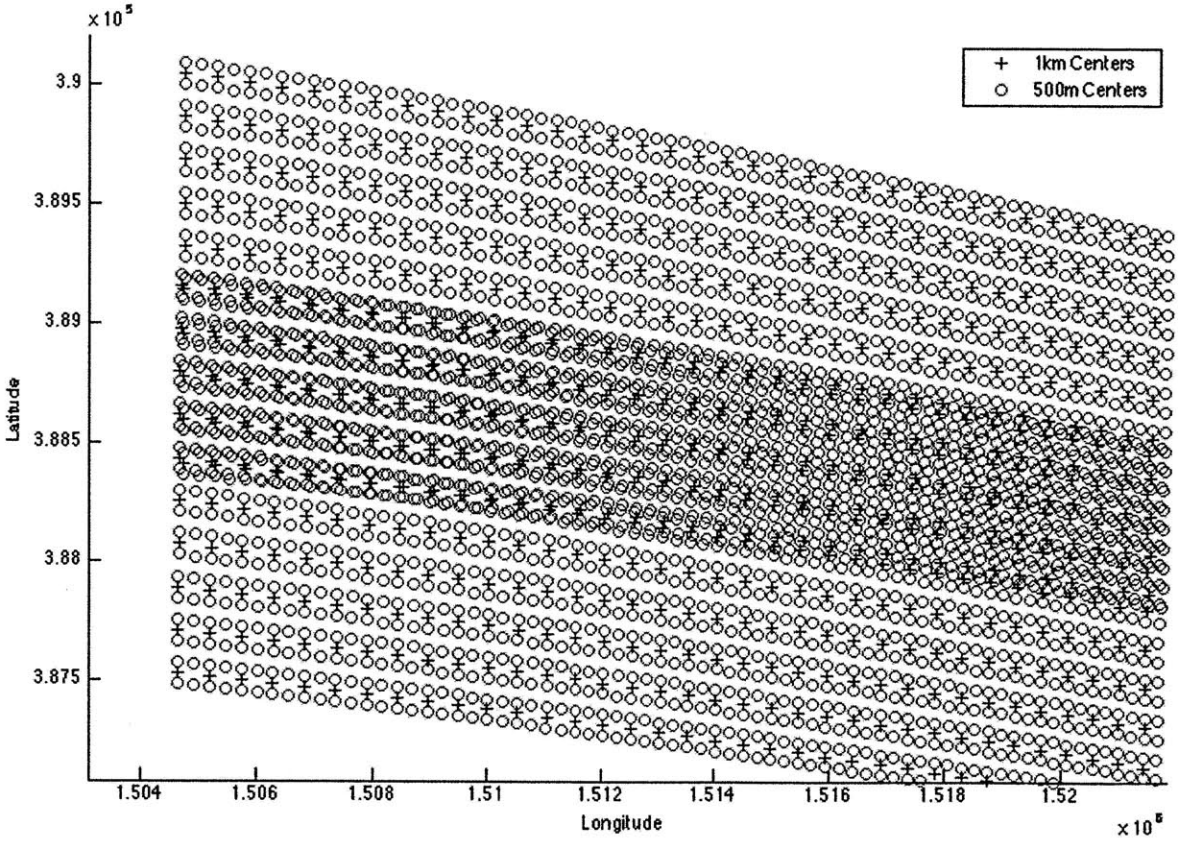

Figure 13: 1km and 500m Pixel Centers Near Start of Scan

Upsampling 1km to 250m

The relationship between 1km pixel centers and 250m pixel centers can be seen visually in Figure 14. In the track direction we see the expected 1:4 relationship, since the triangular weighting functions do not apply in the track direction. In the scan direction, however, we can see the result of the triangular weighting functions in the offset of 250m pixels versus their corresponding 1km pixels. One can also see what appear to be 3 missing 250m pixels at the start of the scans, which were purposefully left out by NASA apparently to preserve a 1:16 data ratio.

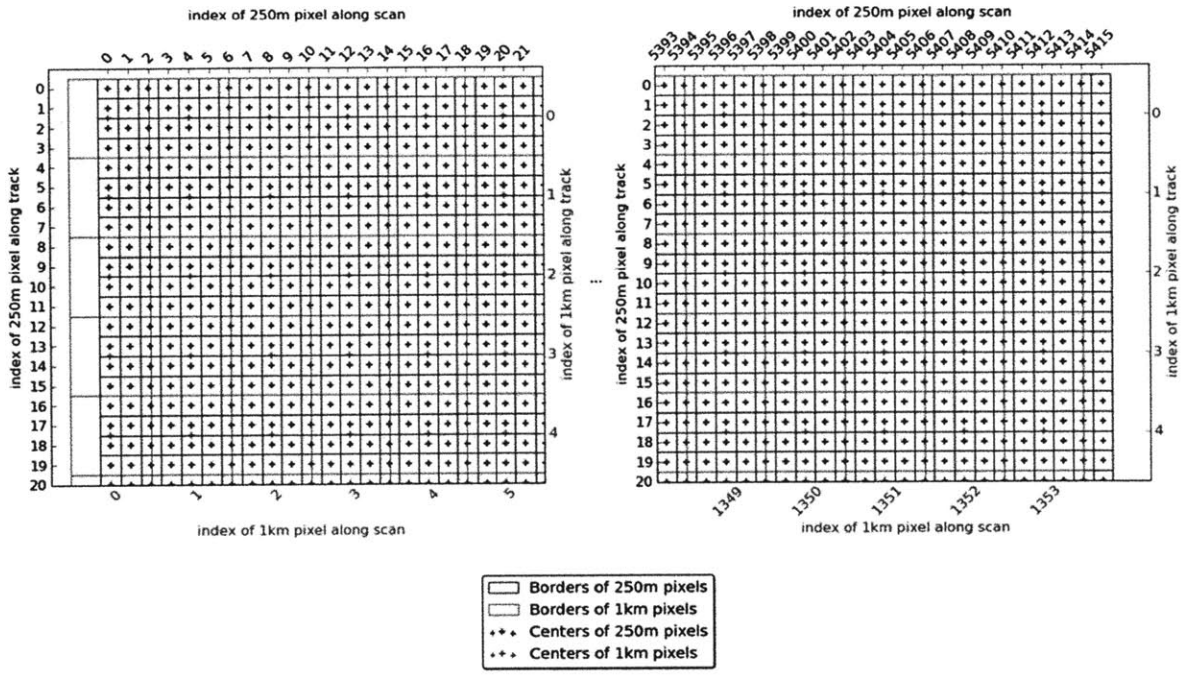

Figure 14: Relative Indices of 1km versus 250m Pixels [31]

For brevity, plots of actual upsampled data are not provided for 250m, as they are very similar to the ones shown in Figure 12 and Figure 13, only denser.

3.4 Field Mappings

This section details the origin and mapping functions used to derive the values for the data fields present in each emitted CSV output file.

3.4.1 Granule_Metadata.csv

This file contains data relevant to the granule as a whole. The data found in the granule metadata (detailed in Table 4) is useful for tracking the source of the data back to the NASA and for debugging. In general, data contained in this file would be used often for analysis. Note that many of the granule metadata fields are embedded into MOD02 files using another encoding called Parameter Value Language (PVL), the specification of which can be found at [32].

Field	Product	Mapping Details
start_time	MOD02	Data Type: int64 Derived From: /CoreMetadata.0/RANGEBEGINNINGDATE/VALUE Derived From: /CoreMetadata.0/RANGEBEGINNINGTIME/VALUE Pattern: yyyyMMddHHmm Mapping: Combine date and time, then apply <Pattern>.
platform_id	MOD02	Data Type: int64 Derived From: /CoreMetadata.0/ASSOCIATEDPLATFORMSHORTNAME/VALUE Mapping: "Aqua" => 0, "Terra" => 1
resolution_id	MOD02	Data Type: int64 Derived From: /CoreMetadata.0/SHORTNAME/VALUE Mapping: "QKM" => 0, "HKM" => 1, "1KM" => 2
scans	MOD02	Data Type: uint8 From: /Number of Scans
track_measurements	MOD02	Data Type: uint16 Derived From: resolution_id Derived From: scans Mapping: {resolution_id: 0 => 40, 1 => 20, 0 => 10} * scans
scan_measurements	MOD02	Data Type: uint16 Derived From: resolution_id Mapping: 0 => 5416, 1 => 2708, 2 => 1354
day_night_flag	MOD02	Data Type: string From: /CoreMetadata.0/DAYNIGHTFLAG/VALUE
file_id	MOD02	Data Type: string From: /CoreMetadata.0/LOCALGRANULEID/VALUE
geo_file_id	MOD02	Data Type: string From: /CoreMetadata.0/ANCILLARYINPUTPOINTER/VALUE

Table 4: Granule_Metadata.csv Field Mappings

3.4.2 Band_Metadata.csv

This file contains values (detailed in Table 5) for deriving measurement values from scaled integers (per band) [33], which are also provided for reference and debugging purposes.

Field	Product	Mapping Details
start_time	MOD02	Inherited from Granule Metadata
platform_id	MOD02	Inherited from Granule Metadata
resolution_id	MOD02	Inherited from Granule Metadata
band_id	MOD02	Data Type: uint8 Prefix: /MODIS_SWATH_Type_L1B/Data Fields Derived From: <Prefix>/EV_1KM_RefSB Derived From: <Prefix>/EV_1KM_Emissive Derived From: <Prefix>/EV_250_Aggr1km_RefSB Derived From: <Prefix>/EV_500_Aggr1km_RefSB Derived From: <Prefix>/EV_500_RefSB Derived From: <Prefix>/EV_250_Aggr500_RefSB Derived From: <Prefix>/EV_250_RefSB Derived From: <Prefix>/EV_Band26 Attribute: band_names (iterate each, keeping track of offset) Mapping: See appendices for band data mappings
radiance_scale	MOD02	Data Type: double From: Same data source as band_id Attribute: radiance_scales (align with corresponding band_names offset)
radiance_offset	MOD02	Data Type: float From: Same data source as band_id Attribute: radiance_offsets (align with corresponding band_names offset)
reflectance_scale	MOD02	Data Type: double From: Same data source as band_id Attribute: reflectance_scales (align with corresponding band_names offset)
reflectance_offset	MOD02	Data Type: float From: Same data source as band_id Attribute: reflectance_offsets
corrected_counts_scale	MOD02	Data Type: double From: Same data source as band_id Attribute: corrected_counts_scales (align with corresponding band_names offset)
corrected_counts_offset	MOD02	Data Type: float From: Same data source as band_id Attribute: corrected_counts_offsets (align with corresponding band_names offset)
specified_uncertainty	MOD02	Data Type: float Prefix: /MODIS_SWATH_Type_L1B/Data Fields From: <Prefix>/EV_1KM_RefSB_Uncert_Indexes From: <Prefix>/EV_1KM_Emissive_Uncert_Indexes From: <Prefix>/EV_250_Aggr1km_RefSB_Uncert_Indexes From: <Prefix>/EV_500_Aggr1km_RefSB_Uncert_Indexes From: <Prefix>/EV_500_RefSB_Uncert_Indexes From: <Prefix>/EV_250_Aggr500_RefSB_Uncert_Indexes From: <Prefix>/EV_250_RefSB_Uncert_Indexes From: <Prefix>/EV_Band26_Uncert_Indexes Attribute: specified_uncertainty (align with corresponding band_names offset)
uncertainty_scaling_factor	MOD02	Data Type: float From: Same source as specified_uncertainty Attribute: scaling_factor (align with corresponding band_names offset)

Table 5: Band_Metadata.csv Field Mappings

3.4.3 Geodata.csv

This file contains geolocation values (detailed in Table 6) from the MOD03 data product, upsampled when applicable. Note that longitude and latitude values in this file are scaled and integerized in preparation for their later use as SciDB dimensions.

Field	Product	Mapping Details
longitude_e4	MOD03	Data Type: int64 Derived From: /MODIS_Swath_Type_GEO/Geolocation Fields/Longitude Mapping: Multiplied by 10000 and rounded. Upsampled when applicable. (Circular domain)
latitude_e4	MOD03	Data Type: int64 Derived From: /MODIS_Swath_Type_GEO/Geolocation Fields/Latitude Mapping: Multiplied by 10000 and rounded. Upsampled when applicable.
start_time	MOD02	Inherited from Granule Metadata
platform_id	MOD02	Inherited from Granule Metadata
resolution_id	MOD02	Inherited from Granule Metadata
track_index	MOD03	Data Type: int16 From: Index of point in data array. Upsampled when applicable.
scan_index	MOD03	Data Type: int16 From: Index of point in data array. Upsampled when applicable.
height	MOD03	Data Type: int16 From: /MODIS_Swath_Type_GEO/Data Fields/Height Upsampled when applicable.
sensor_zenith	MOD03	Data Type: float From: /MODIS_Swath_Type_GEO/Data Fields/SensorZenith Mapping: Divided by 100. Upsampled when applicable.
sensor_azimuth	MOD03	Data Type: float From: /MODIS_Swath_Type_GEO/Data Fields/SensorAzimuth Mapping: Divided by 100. Upsampled when applicable. (Circular domain)
range	MOD03	Data Type: uint32 From: /MODIS_Swath_Type_GEO/Data Fields/Range Mapping: Multiplied by 25. Upsampled when applicable.
solar_zenith	MOD03	Data Type: float From: /MODIS_Swath_Type_GEO/Data Fields/SolarZenith Mapping: Divided by 100. Upsampled when applicable.
solar_azimuth	MOD03	Data Type: float From: /MODIS_Swath_Type_GEO/Data Fields/SolarAzimuth Mapping: Divided by 100. Upsampled when applicable. (Circular domain)
land_sea_mask	MOD03	Data Type: uint8 From: /MODIS_Swath_Type_GEO/Data Fields/Land/SeaMask Upsampled when applicable.

Table 6: Geodata.csv Field Mappings

3.4.4 Band_x_Measurements.csv

This file contains measurement data (detailed in Table 7) for the band specified by "x". Radiance, reflectance and uncertainty percentages are materialized using the scaled integers and corresponding band metadata. This is a selective denormalization that is intended to reduce the need to perform expensive joins and significantly increase analysis performance.

Field	Product	Mapping Details
longitude_e4	MOD03	Inherited from Geodata (aligned by data array index)
latitude_e4	MOD03	Inherited from Geodata (aligned by data array index)
start_time	MOD02	Inherited from Granule Metadata
platform_id	MOD02	Inherited from Granule Metadata
resolution_id	MOD02	Inherited from Granule Metadata
si_value	MOD02	Data Type: uint16 Prefix: /MODIS_SWATH_Type_L1B/Data Fields From: <Prefix>/EV_1KM_RefSB From: <Prefix>/EV_1KM_Emissive From: <Prefix>/EV_250_Aggr1km_RefSB From: <Prefix>/EV_500_Aggr1km_RefSB From: <Prefix>/EV_500_RefSB From: <Prefix>/EV_250_Aggr500_RefSB From: <Prefix>/EV_250_RefSB From: <Prefix>/EV_Band26
radiance	MOD02	Data Type: double Derived From: si_value Dervied From: Band Metadata – radiance_scale Dervied From: Band Metadata – radiance_offset Mapping: radiance_scale * (si_value – radiance_offset) [33]
reflectance	MOD02	Derived From: si_value Dervied From: Band Metadata – reflectance_scale Dervied From: Band Metadata – reflectance_offset Mapping: reflectance_scale * (si_value – reflectance_offset) [33]
uncertainty_index	MOD02	Data Type: uint8 Prefix: /MODIS_SWATH_Type_L1B/Data Fields From: <Prefix>/EV_1KM_RefSB_Uncert_Indexes From: <Prefix>/EV_1KM_Emissive_Uncert_Indexes From: <Prefix>/EV_250_Aggr1km_RefSB_Uncert_Indexes From: <Prefix>/EV_500_Aggr1km_RefSB_Uncert_Indexes From: <Prefix>/EV_500_RefSB_Uncert_Indexes From: <Prefix>/EV_250_Aggr500_RefSB_Uncert_Indexes From: <Prefix>/EV_250_RefSB_Uncert_Indexes From: <Prefix>/EV_Band26_Uncert_Indexes
uncertainty_pct	MOD02	Data Type: float Derived From: uncertainty_index Dervied From: Band Metadata – specified_uncertainty Dervied From: Band Metadata – uncertainty_scaling_factor Mapping: specified_uncertainty * e^(uncertainty_index / uncertainty_scaling_factor) [33]

Table 7: Band_x_Measurements.csv Field Mappings

3.5 Excluded Data

The mod2csv utility does not write all data out to CSV. Aside from data not included in the mappings specified earlier, some data is purposefully not written to CSV. Particularly, longitude values that do not fall within the range -180 to 180, latitude values that doe not fall within the range -90 to 90, measurements having uncertainty index values of 15 or greater, and measurements having scaled integer values greater than 32767 are all excluded since they are invalid/unusable data [**33**].

3.6 Performance Metrics

The mod2csv utility was run one time for each spatial resolution and, during each run, 23 granules were processed into CSV. Table 8 shows the average per-granule metrics relevant to evaluating the overall performance of the mod2csv utility.

Resolution	MOD02 Input Size	MOD03 Input Size	Processing Time	Output Files	Output Size	Output Rate
1km	168.8 MB	31.1 MB	168.7 s	41	5.81 GB	34.4 MB/s
500m	167.6 MB	31.1 MB	123.0 s	10	5.81 GB	47.2 MB/s
250m	184.1 MB	31.1 MB	169.0 s	5	9.55 GB	56.5 MB/s

Table 8: mod2csv Average Performance Metrics Per Granule

For perspective, the total times to process the batches were: 64.7 minutes for the 1km granules, 47.2 minutes for the 500m granules, and 64.8 minutes for the 250m granules. In comparison, it takes approximately 115 minutes for the MODIS sensor to collect the raw data and then additional processing steps are necessary to produce the HDF granules.

3.7 Observations

The mod2csv utility processes all applicable band data in parallel on a per-scan basis. All threads that work in parallel to process a scan must complete before starting to process the next scan. This synchronization is done purposefully to constrain RAM requirements to a level that can run on a modern laptop computer. The higher resolution data also has higher output rates because the processors can do more work per scan and there is less disk contention while writing the CSV output because there are fewer files being written simultaneously.

Faster performance might be realizable if the utility were permitted to make use of more RAM. This additional RAM could be used to buffer more of the data to be read, processed and ultimately written to the band_x_measurements files and make use of a coordinator to maximize disk throughput. The mod2csv utility can, however, already process a large number of MODIS data files across a large cluster completely independently of other instances. Furthermore, once the resulting CSV is generated, there is no need to generate it again unless the exact data required from the MODIS files changes. As a result, the output of the mod2csv utility should minimize rework by providing a checkpoint step.

For a production system that targeted specifically to SciDB, it would be worthwhile to consider building an optimized binary loader that can perform the functionality of the mod2csv utility, but which also can place the resulting data directly into SciDB without producing large CSV intermediate files. This not only would minimize the logistics required to import the MODIS data into SciDB, but also could greatly improve performance. Given the extensibility of SciDB through user-defined functions (UDFs), it is not difficult to extend the database with these capabilities. In fact, a limited HDF/FITS loader has already been created for SciDB [**34**].

Chapter 4: Loading MODIS CSV Data into SciDB

Once mod2csv has generated the CSV output for a set of desired MODIS data granules, the next step toward analysis is to load the resulting CSV data into SciDB. This chapter describes the steps that were undertaken to accomplish that task.

4.1 SciDB Built-In Load Options

Aside from any specialized extensions to SciDB, which may allow for binary loading of particular data format, the database ships with the ability to load two different SciDB-specific text formats: dense load format (DLF) and sparse load format (SLF). Each load format provides a particular way of representing array data. As will be discussed a bit later, in addition to these 2 load formats, SciDB includes a utility called csv2scidb, which enables the loading of CSV data into the database.

DLF is useful when the attributes for every cell in the multidimensional array will be specified and no cells will be empty (though their attributes might be null). With DLF, dimension values are not explicitly written into the load file because they are implicit in the ordering and grouping of the data itself. SLF, on the other hand, is most useful when the data to be imported is sparse and it is more desirable to specify only the attributes for cells that actually have data. When using SLF, the dimensions for each cell must be specified in the load file so that gaps may exist in the data. The choice of which format to use is ultimately dependent on the nature of the data itself. As will be discussed in the next section, both load formats have an inherent requirement that may prove to be burdensome for typical real-world data import projects and directly representing data to be loaded in these formats is not common.

4.2 Loading One-Dimensional Arrays

An impediment to the ease of use of each SciDB load format for multidimensional arrays is the requirement that data in the load files be partitioned and organized by chunk. Meeting this requirement can entail significant preprocessing of the data and forward knowledge of the dimensions and chunk sizing of the target array. This approach may prove infeasible for many real-world big data applications. As a result, an alternative approach illustrated in Figure 15 has been devised, and is currently the recommended way to import data into SciDB: loading data into one-dimensional arrays and then transforming those arrays into higher dimensional ones within SciDB. With the help of the provided csv2scidb utility and a SciDB function called redimension_store, this process can make loading into SciDB from CSV a straightforward task.

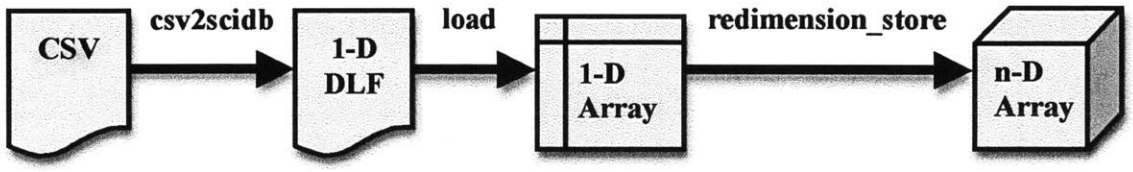

Figure 15: SciDB Recommended Load Process

The csv2scidb utility provided with the SciDB database reads CSV and produces a single, chunked, DLF file suitable for loading into a one-dimensional array. This resulting file can be loaded only from one instance of SciDB, regardless of the number of instances that are members of the cluster. In the next section, a custom-built parallel version of csv2scidb is presented, which allows parallel loading of data across all instances.

4.3 SciDB Parallel Load with pcsv2scidb

As part of this project, a utility that extends the capability of csv2scidb was developed that supports the parallel loading capability of SciDB. The pcsv2scidb utility accepts a parameter specifying the number of instances in the SciDB cluster and rather than emitting a single DLF file, it emits one per instance as shown in Figure 16. The data is distributed to each instance load file via a mechanism that distributes chunks across each instance load file. This approach specifies chunk indices for chunks written into each file in such a way that the chunks loaded by each instance also end up being stored in the instances that loaded them, preventing shuffling of chunks across the network during the load process. Since more instances become involved with the load process and little or no data is shuttled between the instances, the load process using pcsv2scidb should finish faster as the number of instances increases. Benchmarks covered later should help to quantify the benefit of taking advantage of SciDB's parallel loading capabilities.

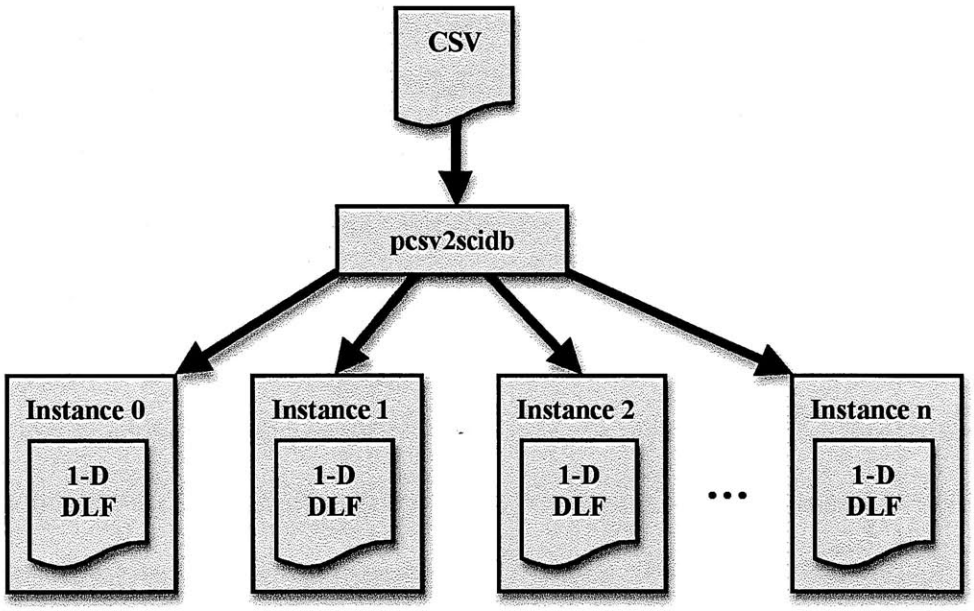

Figure 16: Operation of pcsv2scidb Utility

4.4 Performance Metrics

This section details performance metrics for each aspect of the load process for the same set of granules for which CSV output was created by the mod2csv utility.

4.4.1 Converting CSV to DLF with pcsv2scidb

Table 9 shows the average per-granule metrics for converting CSV to one-dimensional DLF for a range of 1 to 5 instances and over each of the 3 spatial resolutions. Note that all CSV files for each granule were processed in parallel up to 9 at a time, which could be responsible for some disk contention and performance fluctuations. This approach was experimentally found faster than processing each file serially and was used instead. The metrics presented in this section do not include loading of data into SciDB, but rather only the transformation of the input CSV into the one-dimensional DLF format that is suitable for loading into one-dimensional load arrays.

Instances	Resolution	Input/Output Files	Input CSV Size	Processing Time	Output DLF Size	Output Rate
1	1km	41	5.81 GB	64.7 s	6.08 GB	94.0 MB/s
	500m	10	5.81 GB	60.6 s	6.06 GB	100.0 MB/s
	250m	5	9.55 GB	102.7 s	9.94 GB	96.8 MB/s
2	1km	41	5.81 GB	64.6 s	6.08 GB	94.1 MB/s
	500m	10	5.81 GB	60.8 s	6.06 GB	99.7 MB/s
	250m	5	9.55 GB	109.5 s	9.94 GB	90.8 MB/s
3	1km	41	5.81 GB	65.1 s	6.08 GB	93.4 MB/s
	500m	10	5.81 GB	61.7 s	6.06 GB	98.2 MB/s
	250m	5	9.55 GB	118.4 s	9.94 GB	84.0 MB/s
4	1km	41	5.81 GB	66.4 s	6.08 GB	91.6 MB/s
	500m	10	5.81 GB	64.0 s	6.06 GB	94.7 MB/s
	250m	5	9.55 GB	102.3 s	9.94 GB	97.2 MB/s
5	1km	41	5.81 GB	67.9 s	6.08 GB	89.5 MB/s
	500m	10	5.81 GB	65.9 s	6.06 GB	91.9 MB/s
	250m	5	9.55 GB	91.1 s	9.94 GB	109.1 MB/s

Table 9: pcsv2scidb Average Performance Metrics Per Granule

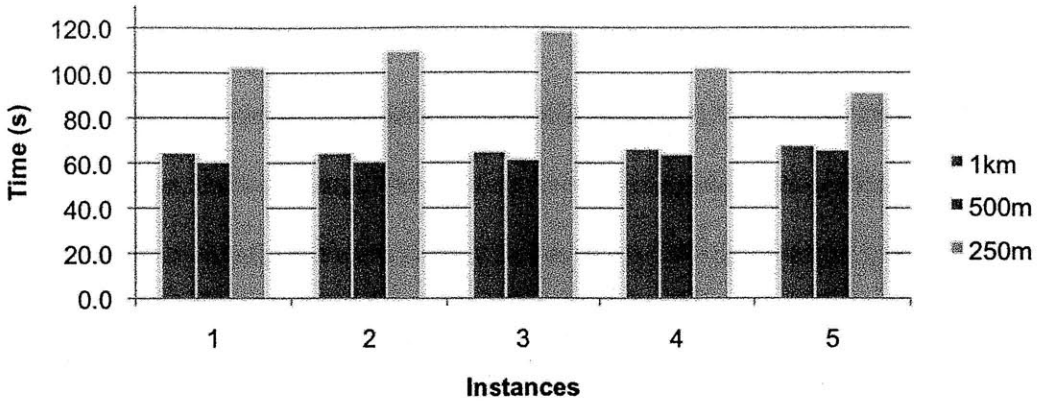

Figure 17: pcsv2scidb Average Processing Time Per Granule

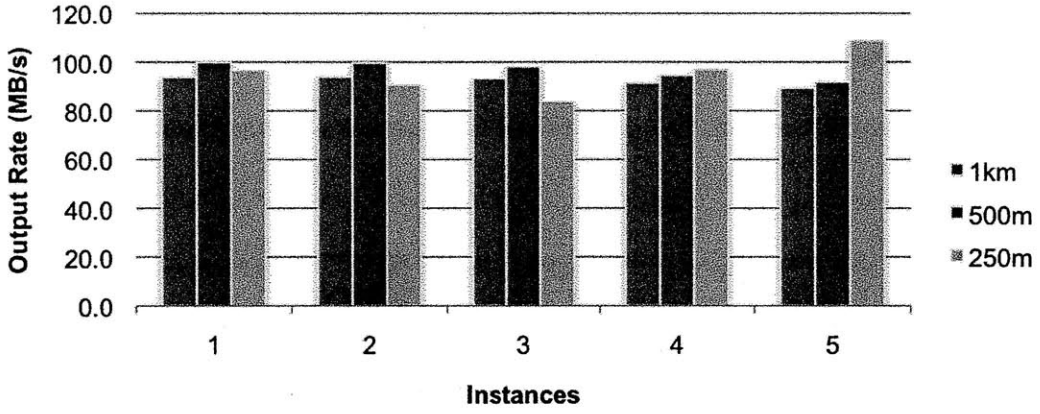

Figure 18: pcsv2scidb Average Output Rate Per Granule

With the exception of some small fluctuations, Table 9, Figure 17, and Figure 18, show that the performance of the pcsv2scidb utility overall holds fairly steady. Performance at each spatial resolution is a factor of the average amount of data in each CSV file comprising a granule and the total number of such files. Larger processing times for the 250m granules are directly attributable to the larger overall size of the granules. We can see from Figure 18 that the output rates for all 3 spatial resolutions tend to remain within a somewhat narrow margin.

4.4.2 One-Dimensional DLF Load

Unsurprisingly, loads are performed in SciDB using the "load" operation. For reasons discussed earlier, data for this project is imported into SciDB using a two-step process: parallel load data into one-dimensional arrays and then re-dimension them into their multidimensional analysis arrays. This section presents benchmarking results for the first of those two steps over local (same machine) clusters of 1 to 5 instances and across all 3 spatial resolutions. Note that there is a one-to-one mapping between the DLF load files and the one-dimensional load arrays in SciDB.

Instances	Resolution	Input Files	Input Size	Concurrency	Load Time	Load Arrays	Load Rate
1	1km	41	6.08 GB	6	236.4 s	41	25.7 MB/s
	500m	10	6.06 GB	4	256.2 s	10	23.7 MB/s
	250m	5	9.94 GB	2	772.9 s	5	12.9 MB/s
2	1km	41	6.08 GB	6	102.2 s	41	59.5 MB/s
	500m	10	6.06 GB	4	130.6 s	10	46.4 MB/s
	250m	5	9.94 GB	2	453.5 s	5	21.9 MB/s
3	1km	41	6.08 GB	6	85.1 s	41	71.4 MB/s
	500m	10	6.06 GB	4	91.6 s	10	66.2 MB/s
	250m	5	9.94 GB	2	311.3 s	5	31.9 MB/s
4	1km	41	6.08 GB	6	82.4 s	41	73.8 MB/s
	500m	10	6.06 GB	4	82.3 s	10	73.6 MB/s
	250m	5	9.94 GB	2	250.9 s	5	39.6 MB/s
5	1km	41	6.08 GB	6	74.9 s	41	81.2 MB/s
	500m	10	6.06 GB	4	77.6 s	10	78.1 MB/s
	250m	5	9.94 GB	2	272.9 s	5	36.4 MB/s

Table 10: Average One-Dimensional DLF Load Metrics Per Granule

The concurrency column of Table 10 indicates the number of DLF files that were imported in parallel in this benchmark. These concurrency values have been purposefully varied between spatial resolutions to reduce swapping because file sizes vary between spatial resolutions, which directly impacts memory utilization.

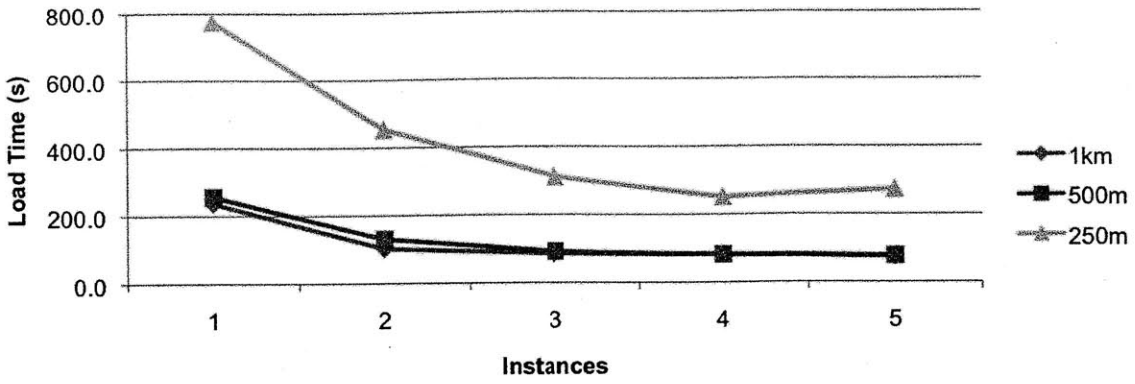

Figure 19: 1-D Load Time by Number of Instances

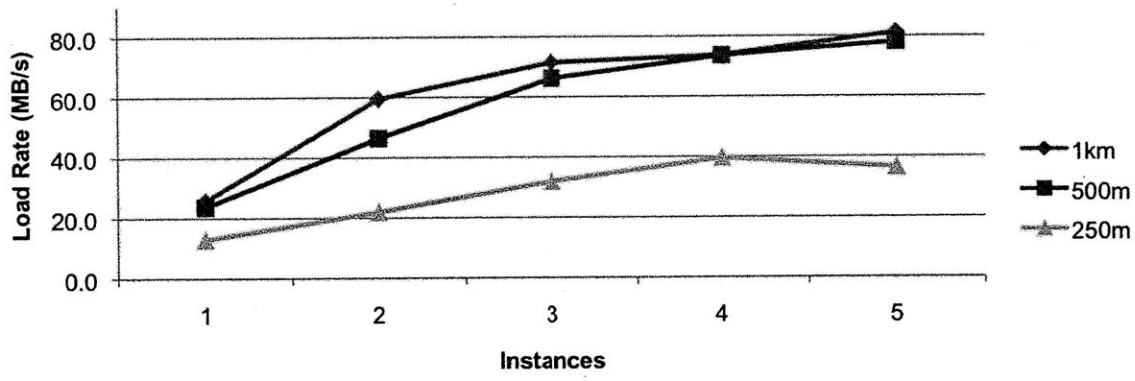

Figure 20: 1-D Load Rate by Number of Instances

From Figure 19 and Figure 20, we can see that load time decreases and load rate generally increases as we add additional instances to the cluster. One exception is for 250m data being loaded into 5 instances. The server began to experience contention at this level of parallelism that was parasitic on the performance. This would not be the case in a true cluster of physically separate servers. Theoretically the load time should converge asymptotically to a number slightly larger than the time required to load a single chunk into each instance if the number of instances was equal to the number of chunks being loaded.

4.4.3 Re-Dimension Store

The final step to loading data into SciDB is to re-dimension the one-dimensional load data into the multidimensional analysis arrays using the redimension_store operation. This section presents the metrics collected during that second load stage.

Instances	Resolution	Input/Output Arrays	Concurrency	Processing Time
1	1km	41	6	98.7 s
	500m	10	4	155.8 s
	250m	5	2	504.1 s
2	1km	41	6	83.4 s
	500m	10	4	134.9 s
	250m	5	2	470.3 s
3	1km	41	6	83.1 s
	500m	10	4	130.6 s
	250m	5	2	469.4 s
4	1km	41	6	78.6 s
	500m	10	4	133.6 s
	250m	5	2	673.5 s
5	1km	41	6	93.7 s
	500m	10	4	196.1 s
	250m	5	2	2,171.1 s

Table 11: Re-Dimension Store Metrics Per Granule

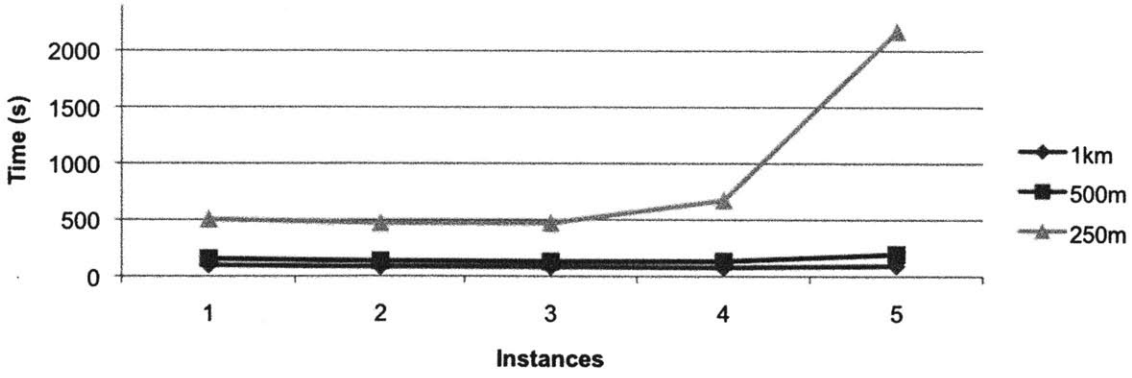

Figure 21: Re-Dimension Store Time Per Granule by Number of Instances

From Table 11 and Figure 21, we can see that the time to perform the redimension_store operation increased dramatically after increasing the cluster size from 3 to 5 instances. The reason for this is that the benchmark server was observed to run out of RAM resources and began swapping heavily. Notably, the SciDB system was able to complete the operation eventually despite running out of RAM in the process.

Aside from the results for the 4-5 instances, the smaller cluster sizes did show a benefit from parallelization of the redimension_store operation. This benefit, however, was not nearly as dramatic for the load operation in this case because the nature of the operation limits its ability to scale horizontally. Unfortunately, it appears as though the redimension_store has to be thought of as a nearly a constant-time operation despite the number of cluster instances.

4.4.4 MODBASE Schemas

The MODBASE schemas can be found in the appendices of this document. As was mentioned earlier, there are two schemas: one-dimensional load schema and analysis schema. The one-dimensional load schema is transient and created only during loads by the load tools themselves. Once data has been re-dimensioned into the analysis arrays, the load arrays are removed. The analysis schema is designed to optimize performance of typical Earth Science queries. Each bands is stored in its own array because many operation operate only on small subsets of bands. The SciDB join operation makes it trivial to efficiently combine multiple bands together, side-by-side. Also, by de-normalizing the bands into separate arrays, these arrays can use the SciDB join operation to join on the geodata array as well. This allows very efficient queries to be authored, which can filter data based on attributes such as sensor zenith/azimuth.

Chapter 5: Earth Science Benchmarks

This chapter presents benchmarks that help both qualitatively show how SciDB can be used to perform basic analysis pertinent to the Earth Science community and quantitatively demonstrate the time needed to accomplish such analysis on the specified benchmark server. All benchmarks in this chapter have been run using 500m MODIS data exclusively as it is the highest resolution having data for all the bands needed to run each of the benchmarks. Source code used to perform these benchmarks has been provided in the appendices for reference and is implemented as a series of database queries driven by Bash shell scripts.

While the resolution of the data used for the benchmarks was held constant for all the benchmarks, the number of instances and the size of the data window to be processed were varied to help paint a picture of how effectively SciDB can parallelize work across an increasing number of instances and data. The number of instances was varied between 1 and 5. Through experimentation, it was found that a 5 instances cluster was pushing the memory limits of the benchmarking server on certain tasks. Three data windows were used by each benchmark. Each data window is a geographic subset (filtered on latitude and longitude) of the data that was loaded into the database. The three windows are 10° x 10°, 20° x 20°, and 30° x 30° in size, centered over an area having a high concentration of data points.

To help provide additional insight into benchmark performance beyond just the processing time, the number of cells processed by each benchmark has also been captured and the number of cells processed per second has been provided. This additional metric helps to normalize the performance as a function of the amount of work being done per unit time by the benchmark. Note, all benchmarks were run 3 times and the results were averaged.

5.1 RGB Composite Images

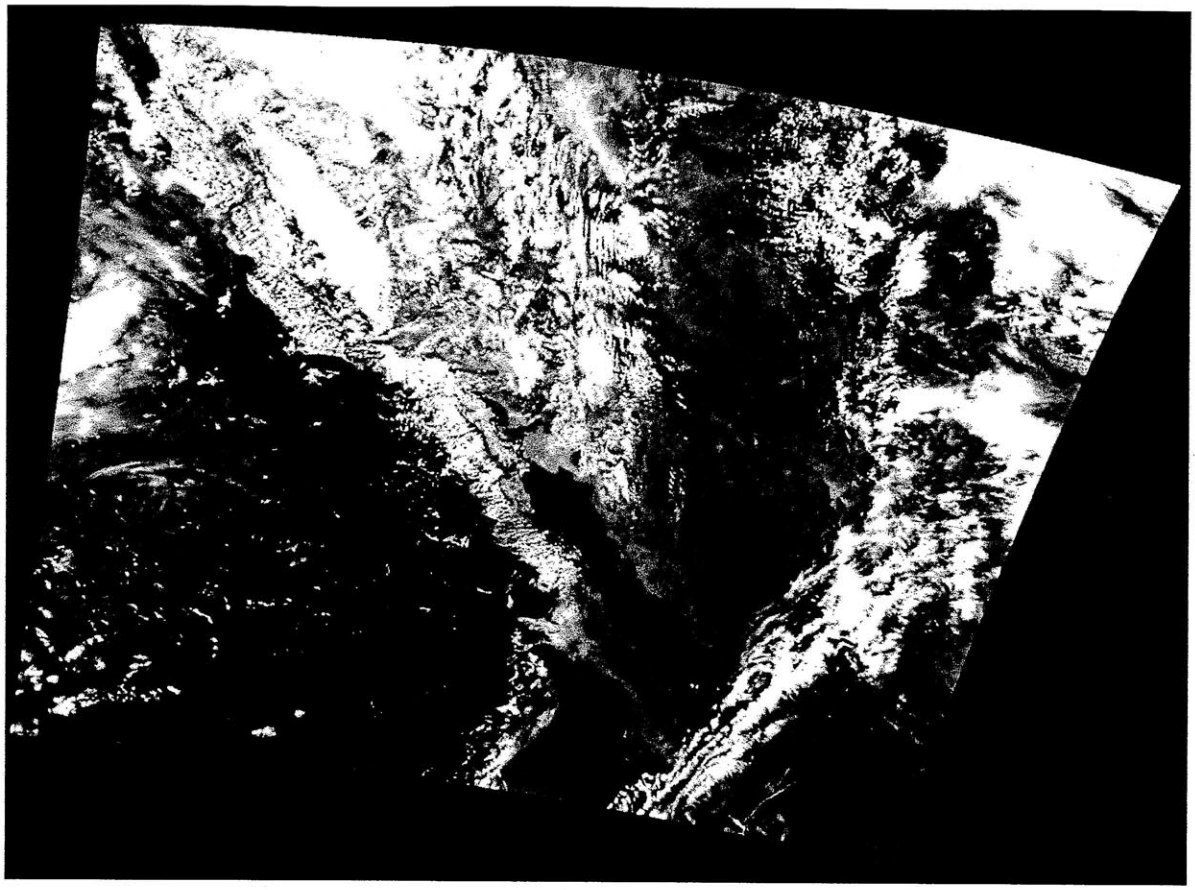

Figure 22: RGB Composite Image of a Single 500m MODIS Granule

Generating red, green, and blue (RGB) composite images from MODIS data is a relatively simple, but useful, task that exercises important core properties of the SciDB database. RGB composite images present MODIS data as a human viewing the Earth from the satellite's perspective might see it. Figure 22 is an example of an RGB composite image generated as a result of gridding and visualizing data generated from this benchmark. Three separate band data sets must be accessed, filtered and combined but no further processing is performed. The

MODIS bands that most directly correspond to red, green and blue reflectance values are bands 1 (red), 4 (green) and 3 (blue). The resulting metrics are shown in Table 12 and graphed in Figure 23 and Figure 24.

Instances	Window Size	Cells	Processing Time	Cells / Second
1	10° x 10°	3,511,861	30.9 s	113,775
	20° x 20°	9,086,027	68.1 s	133,357
	30° x 30°	10,994,294	79.9 s	137,658
2	10° x 10°	3,511,861	20.1 s	174,719
	20° x 20°	9,086,027	36.5 s	249,160
	30° x 30°	10,994,294	42.3 s	260,117
3	10° x 10°	3,511,861	16.4 s	213,704
	20° x 20°	9,086,027	30.4 s	298,882
	30° x 30°	10,994,294	33.9 s	324,315
4	10° x 10°	3,511,861	12.2 s	288,646
	20° x 20°	9,086,027	26.2 s	347,237
	30° x 30°	10,994,294	28.2 s	389,408
5	10° x 10°	3,511,861	22.7 s	154,708
	20° x 20°	9,086,027	34.3 s	265,156
	30° x 30°	10,994,294	36.8 s	299,029

Table 12: RGB Composite Benchmark Metrics

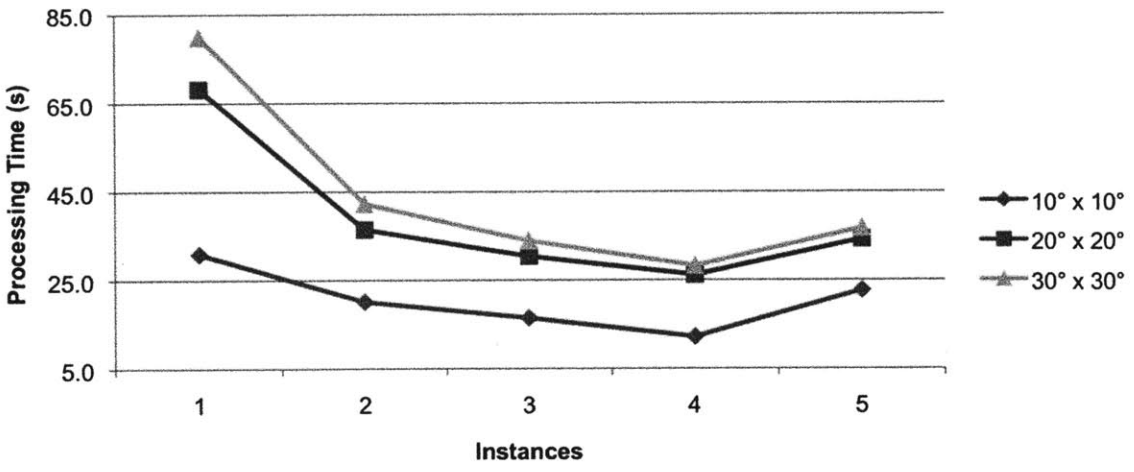

Figure 23: Average RGB Composite Processing Time by Number of Instances

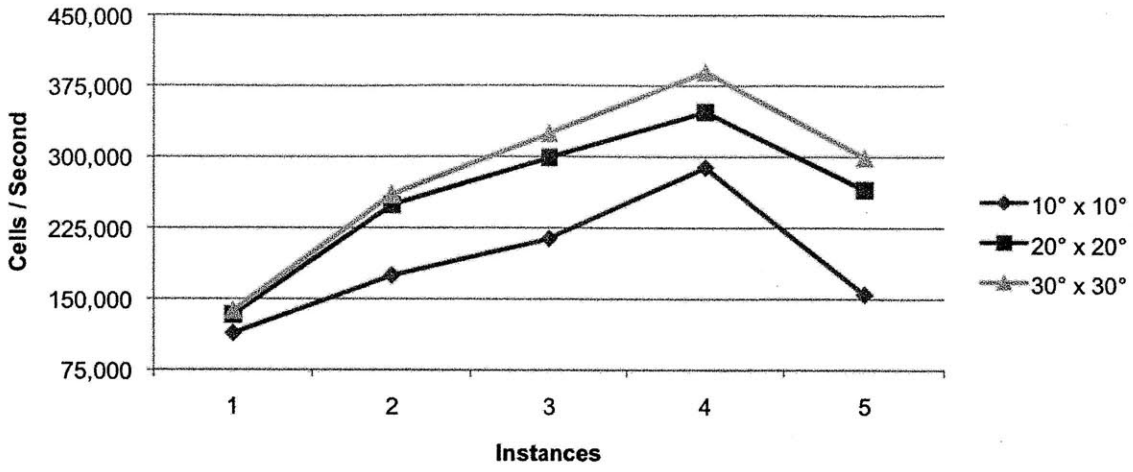

Figure 24: Average RGB Composite Cells / Second by Number of Instances

From these metrics and graphs, it is immediately apparent that we have again encountered the same problem that we encountered during redimension_store when the cluster sized was increased from 4 to 5 instances. The benchmarking server ran out of RAM resources and experienced swapping that negatively affected performance. From 1 to 4 instances, however, performance did increase as expected with the addition of new instances.

Though it is intuitive that processing more data could take more time than processing less, in an ideal world the cells / second that are processed would be the same regardless of the number of input cells. However, given the discrete nature of the SciDB storage and processing engine, which is chunk-based, the possibility of an uneven distribution of chunks across instances, and the likelihood that not all chunks will have the same cell density, divergence in performance such as the one see in Figure 24 are probable. As the data window (a subset of the array dimensions) increases in size, there are more chunks that may be processed in parallel, which might partly explain why larger data windows exhibit better performance.

5.2 Normalized Difference Vegetation Index (NDVI)

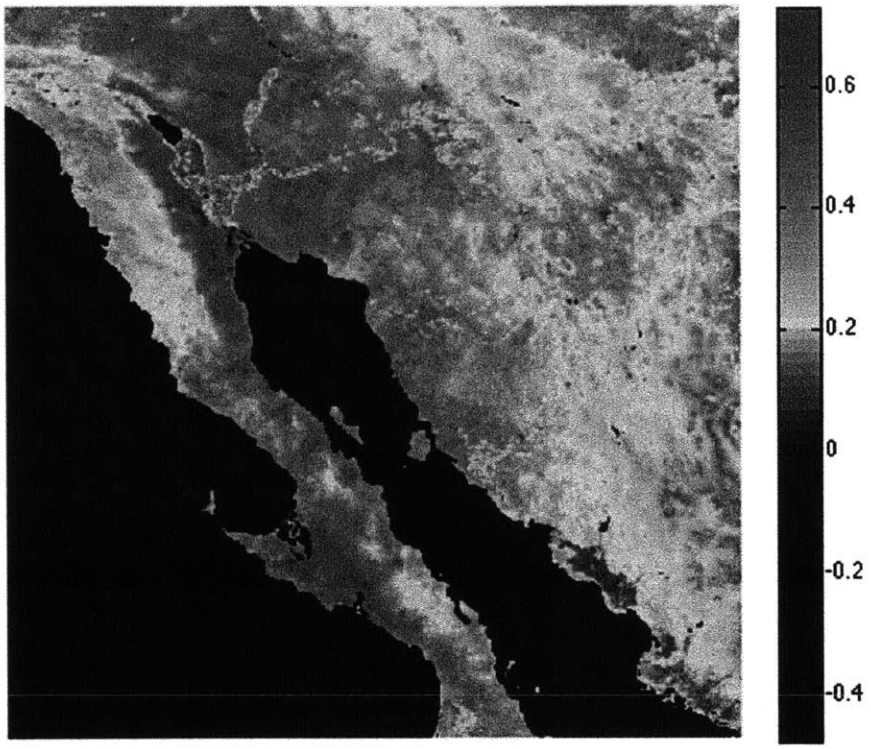

Figure 25: NDVI Image Created from a portion of a 500m MODIS Granule

The NDVI is a common Earth Science analysis that uses the reflectance properties of vegetation in the red and near-infrared (nir) bands to identify the relative concentration of vegetation in a given area. NDVI = (nir – red) / (nir + red). For MODIS, band 1 corresponds to red and band 2 corresponds to nir. NDVI can be run on 250m resolution data since data for both bands is provided at that resolution. For consistency, however, this benchmark was run on 500m data. Table 13, Figure 26, and Figure 27 present the performance metrics for the NDVI benchmark.

Instances	Window Size	Cells	Processing Time	Cells / Second
1	10° x 10°	25,105,811	147.2 s	170,517
	20° x 20°	86,482,367	462.9 s	186,827
	30° x 30°	151,772,170	784.5 s	193,455
2	10° x 10°	25,105,811	87.2 s	287,801
	20° x 20°	86,482,367	283.2 s	305,376
	30° x 30°	151,772,170	477.0 s	318,181
3	10° x 10°	25,105,811	80.0 s	313,953
	20° x 20°	86,482,367	254.9 s	339,324
	30° x 30°	151,772,170	399.6 s	379,810
4	10° x 10°	25,105,811	68.2 s	367,941
	20° x 20°	86,482,367	203.0 s	426,091
	30° x 30°	151,772,170	381.9 s	397,413
5	10° x 10°	25,105,811	66.2 s	379,433
	20° x 20°	86,482,367	189.6 s	456,131
	30° x 30°	151,772,170	347.7 s	436,461

Table 13: NDVI Benchmark Metrics

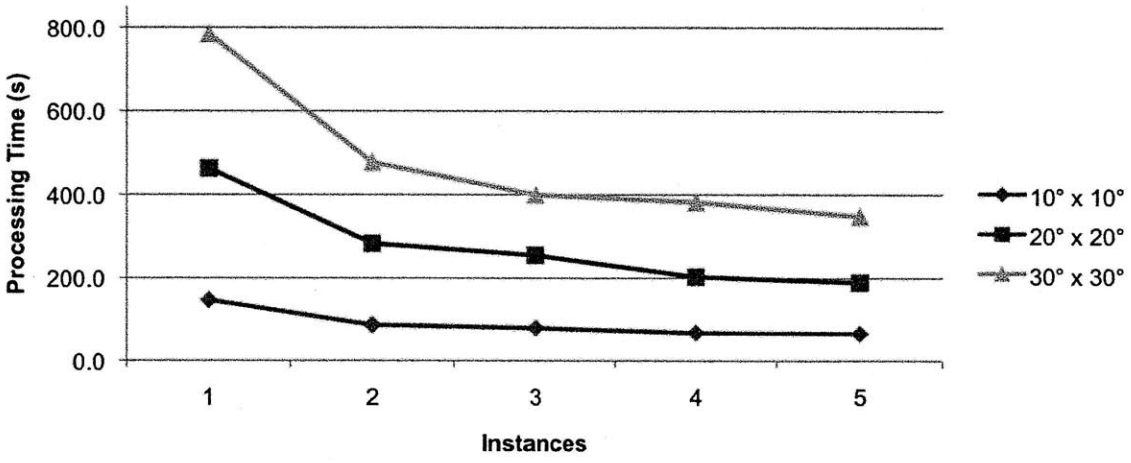

Figure 26: Average NDVI Processing Time by Number of Instances

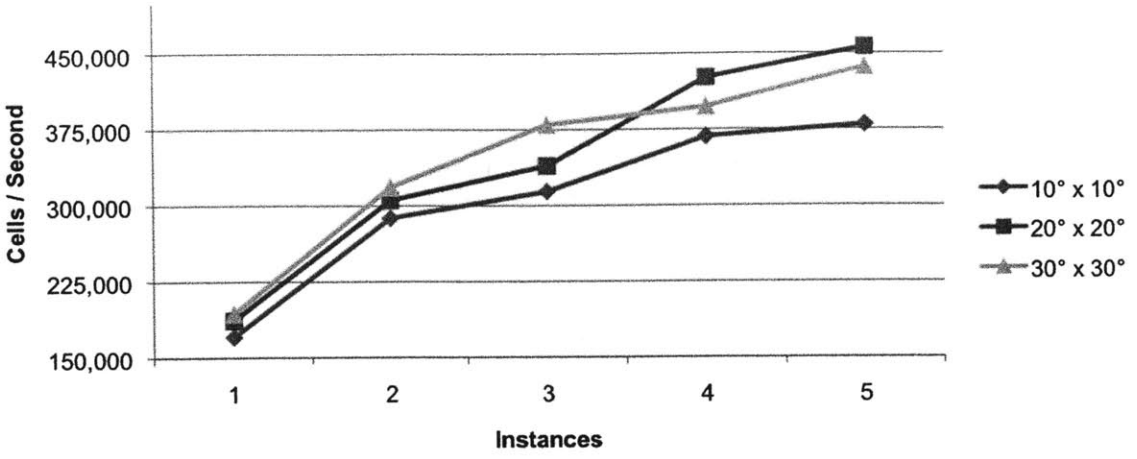

Figure 27: Average NDVI Cells / Second by Number of Instances

The results of this benchmark show that the NDVI calculation also exhibits significant performance benefits from having more instances in the cluster. Like the prior benchmark, we see that the cells/second processed on a single instance are closely grouped for each of the window sizes, but that some divergence is noticeable as the number of instances in the cluster is increased. In general, larger data windows still seem slightly outperform the smaller ones, with an exception in this case of the 20° x 20° overtaking the 30° x 30° for 4 and 5 instances.

Despite adding calculations, this benchmark performs very strongly relative to the RGB composite benchmark, which has joins instead. It is interesting to note that the incremental benefit of adding additional instances to the cluster appears to decrease with each added instance. This makes sense from a strictly mathematical examination, however, as each additional instance represents 1/n of the total capacity of the cluster. As n gets larger, the relative contribution of each additional instance gets proportionally smaller. Furthermore, as we add more instances to the same physical machine, they start to compete with each other for limited resources.

5.3 Normalized Difference Snow Index (NDSI) + Snowmap

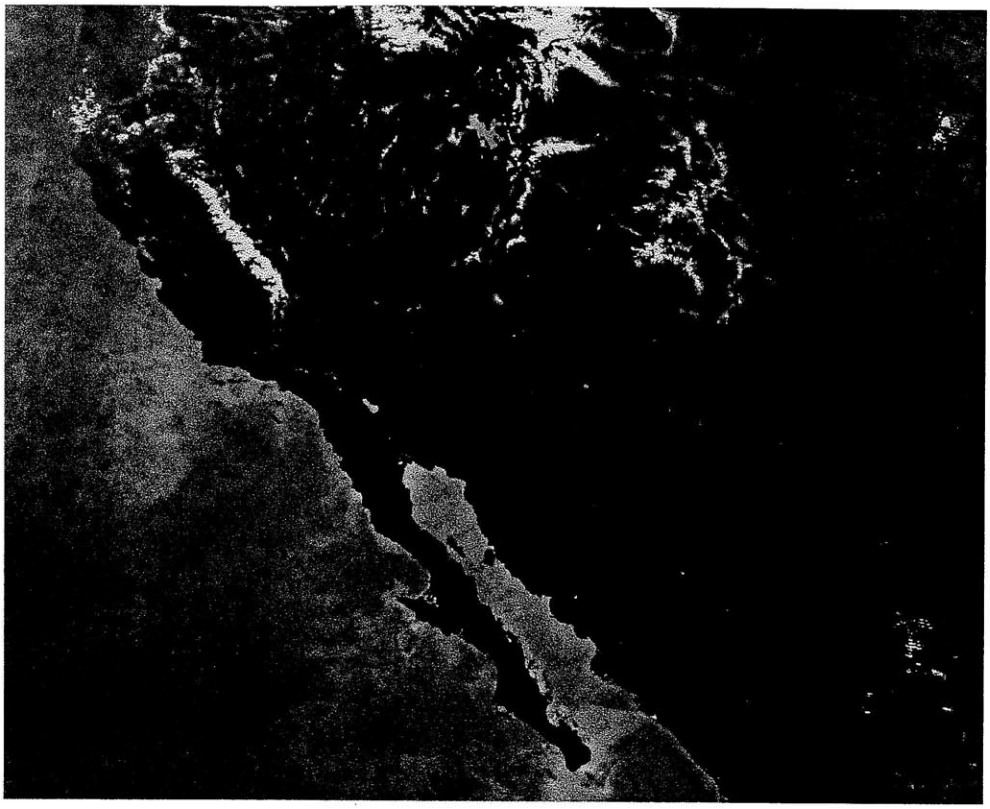

Figure 28: NDSI+Snowmap Image Created from a portion of a 500m MODIS Granule

The NDSI and Snowmap provide a means of determining where snow cover in a given scene. The NDSI is very similar to the NDVI, with the exception that it uses bands 4 and 6 rather than 1 and 2. NDSI = (Band 4 – Band 6) / (Band 4 + Band 6). The NDSI, however, sometimes has trouble discriminating snow from water or from certain types of vegetation under certain circumstances [**35**]. The Snowmap is a binary indicator of snow that is derived from the positive outcome on three separate tests: NDSI > 0.4, Band 2 > 0.11 and Band 4 > 0.1. An additional lowering of the NDSI threshold is sometimes done in areas with high NDVI, but this test was not

added to this benchmark. Table 14, Figure 29, and Figure 30 present the performance metrics for the NDSI+Snowmap benchmark.

Instances	Window Size	Cells	Processing Time	Cells / Second
1	10° x 10°	22,357,813	288.5 s	77,497
	20° x 20°	78,252,073	942.3 s	83,047
	30° x 30°	139,088,124	1,608.5 s	86,472
2	10° x 10°	22,357,813	164.6 s	135,831
	20° x 20°	78,252,073	536.4 s	145,893
	30° x 30°	139,088,124	939.3 s	148,082
3	10° x 10°	22,357,813	155.8 s	143,473
	20° x 20°	78,252,073	433.1 s	180,665
	30° x 30°	139,088,124	738.2 s	188,407
4	10° x 10°	22,357,813	109.0 s	205,055
	20° x 20°	78,252,073	349.7 s	223,790
	30° x 30°	139,088,124	624.0 s	222,886
5	10° x 10°	22,357,813	113.9 s	196,351
	20° x 20°	78,252,073	306.5 s	255,309
	30° x 30°	139,088,124	571.7 s	243,303

Table 14: NDSI+Snowmap Benchmark Metrics

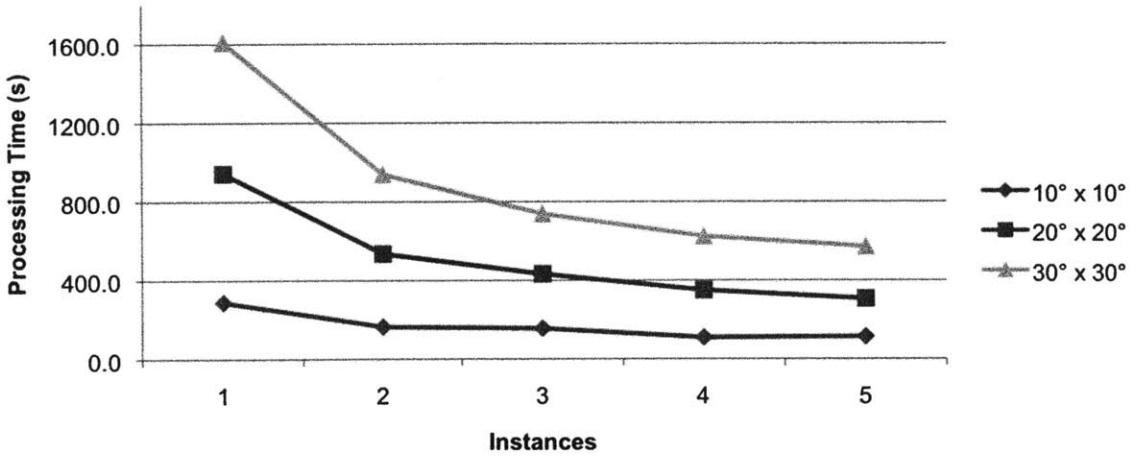

Figure 29: Average NDSI+Snowmap Processing Time by Number of Instances

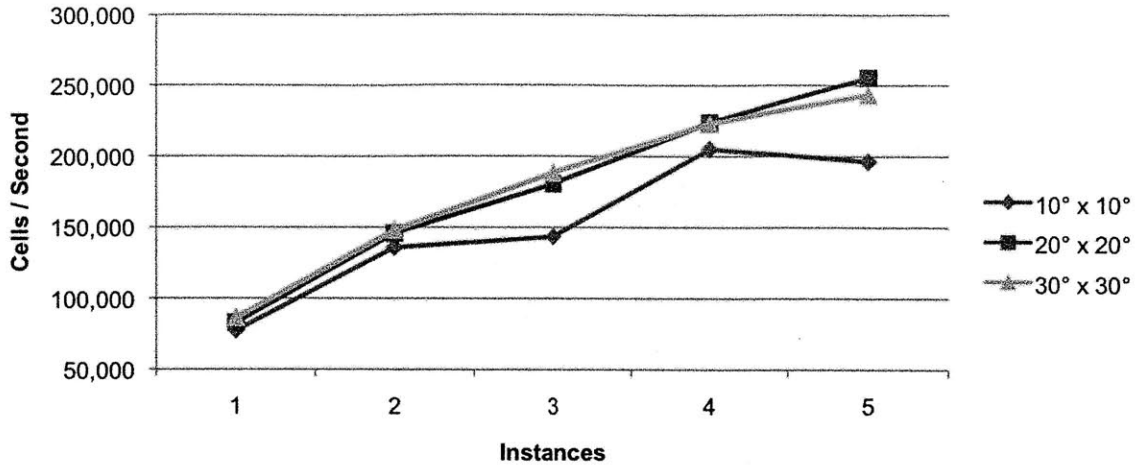

Figure 30: Average NDSI+Snowmap Cells / Second by Number of Instances

The performance characteristics of the NDSI+Snowmap show stark similarities to the characteristics of the NDVI benchmark, despite taking nearly twice as long to complete (because it is doing roughly twice the work). The performance of 30° x 30° and 20° x 20° windows closely track each other initially and cross over at 4 instances, as they did in the NDVI benchmark. Additionally, the 10° x 10° window also seems to plateau at 3 and 5 instances, as was the case in the NDVI benchmark (though this time it slid backward a bit in performance). For the larger data windows, it appears that each additional instance continued to show performance gains and another few instances would likely continue that trend. The performance "stalls" seen in the 10° x 10° to some extent in the NDVI benchmark and to a greater extent in this benchmark are curious and deserve further study. They may have some relation to the distribution of chunks across the cluster or the uneven distribution of data within chunks leading to the particular chunks involved in this benchmark being unevenly distributed.

5.4 Gridding Data

Gridding data is a very task that is used to down sample data for the purposes of comparison or display. Attempting to plot 150 million sparse pixels would generally yield unsatisfactory results. Instead, aggregation functions are used to create a single pixel value in the grid space from potentially many pixels in a higher-resolution space. SciDB has a built in function to exactly this operation called "regrid". The regrid operation would be a perfect choice if all pixels can be treated as point sources. However, because MODIS pixels actually represent data collected over a significant spatial area, representing the as point sources may not be desirable. To show that SciDB can accommodate this use case, another technique that amounts to performing aggregations using overlapping windows was devised. The amount of overlap for each window corresponds to roughly half of a MODIS pixel, allowing influencing pixels to be incorporated that would have otherwise been excluded by the regrid operation.

This benchmark measures the performance of the devised method for allowing overlapping windows rather than that of regrid, which is certainly much more efficient. Because SciDB currently does not currently have explicit support for an overlapping regrid operation, or a window operation with start/step syntax, this capability had to be emulated using other SciDB operations. Though it works as expected, this method of gridding is computationally expensive and could easily be optimized through the creation of a native operation. Given the computationally intensive nature of the work being performed, or all the benchmarks run in this project, the gridding benchmark is the most challenging for the database to perform. Table 15, Figure 31, and Figure 32 present the performance metrics for the gridding benchmark.

Instances	Box Size	Cells	Processing Time	Cells / Second
1	10° x 10°	25,105,811	506.2 s	49,592
	20° x 20°	86,482,367	1,556.4 s	55,567
	30° x 30°	151,772,170	2,633.5 s	57,631
2	10° x 10°	25,105,811	318.9 s	78,735
	20° x 20°	86,482,367	1,114.4 s	77,604
	30° x 30°	151,772,170	1,876.3 s	80,888
3	10° x 10°	25,105,811	208.4 s	120,489
	20° x 20°	86,482,367	743.5 s	116, 318
	30° x 30°	151,772,170	1,269.8 s	119,528
4	10° x 10°	25,105,811	189.1 s	132,788
	20° x 20°	86,482,367	626.9 s	137,960
	30° x 30°	151,772,170	1,051.9 s	144,279
5	10° x 10°	25,105,811	169.6 s	148,030
	20° x 20°	86,482,367	577.2 s	149,831
	30° x 30°	151,772,170	981.9 s	154,565

Table 15: Gridding Benchmark Metrics

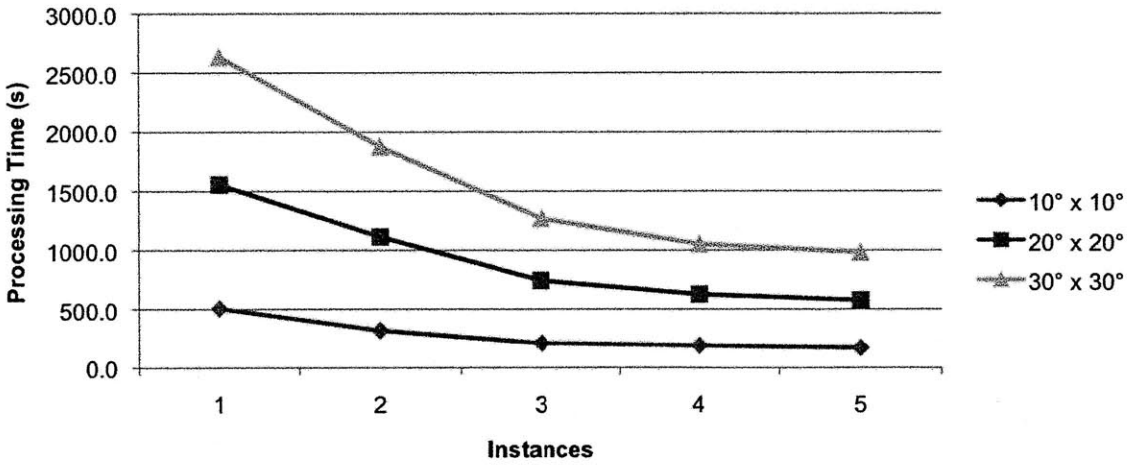

Figure 31: Average Gridding Processing Time by Number of Instances

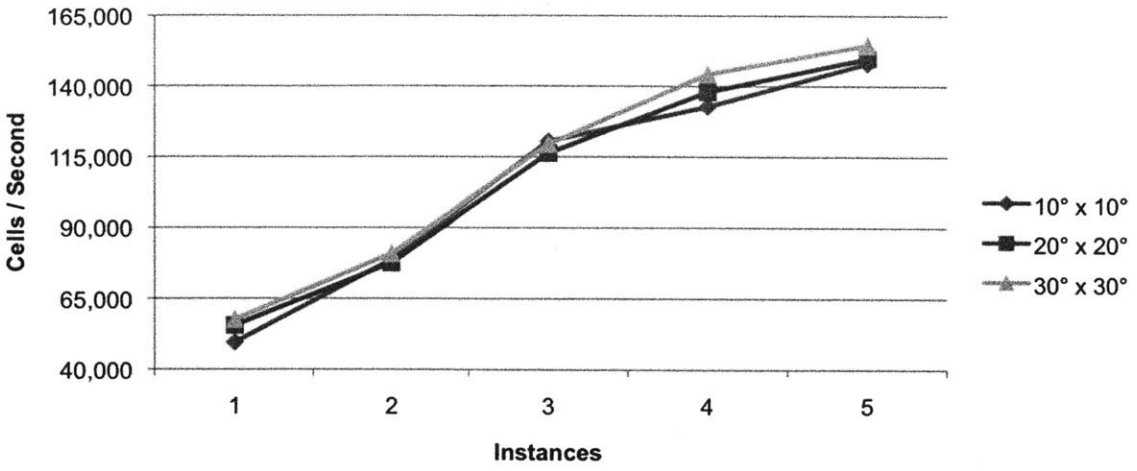

Figure 32: Average Gridding Cells / Second by Number of Instances

This benchmark displays the same characteristic curves we saw from the prior ones, except that the deviations in the performance of each of the data windows from the perspective of cells/second processed are much smaller. As with the others, there are marginal increases in performance as additional instances are added and there appear to be performance asymptotes to which each plot appears to be converging. We can see, however, that this is a computationally expensive benchmark from the time needed to process it and the lower cells/second processed.

Chapter 6: Conclusions and Future Work

The primary objective of this project was to design, implement, and characterize the performance of a system for analyzing MODIS data, built around and upon the SciDB multidimensional array database. In short, MODBASE has met all of the stated objectives. A preprocessor component was built to extract and upsample MODIS HDF-EOS data and a parallel loader component was created to increase the ingest rate of the system by taking advantage of SciDB's ability to perform distributed loading across all cluster instances. One-dimensional load and multidimensional analysis schemas were designed and data was loaded through a CSV to SciDB load process. Once the data was loaded, detailed performance metrics were taken over a cross product of 4 benchmarks x 5 clusters x 3 data windows = 60 configurations (each of which were run 3 times and averaged). In all cases, it was clear that SciDB was making use of additional instances to spread not only the storage burden, but also the computational burden. The MODBASE system shows great promise for scalability and provides an intuitive platform for performing ad-hoc analysis via a query language interface supporting multidimensional array operations that perfectly suit the Earth Science domain.

For 3 instances installed on the commodity benchmark server used for this project, the total time to preprocess and load a single granule of data varied between approximately 5-6 minutes for 1km and 500m granules and 20 minutes for 250m granules. Adding more physical servers to the cluster would decrease the total time needed to preprocess HDF granules while increasing the number of granules that could be processed in parallel. A cluster with sufficient capacity should therefore be capable of loading the entire multi-petabyte MODIS Level-1B data set and incrementally incorporating new granules as they become available. The resulting data

store could then be analyzed in an ad-hoc fashion using queries such as the ones used to perform the benchmarks for this project. Analytical results can be persisted to new arrays, which themselves can be further analyzed. Entire analysis pipelines can be constructed and automated, allowing for regularly scheduled creation of an ecosystem of analytical products. The benchmarks performed in this project indicate that, provided cluster instances do not run out of memory or processing resources, the aggregate number of cells per second that can be processed by clusters of increasing size also increases proportionally.

As is common with most database systems, loading data into SciDB is time consuming and warrants further investigation. The approach taken in the course of this project was the one most commonly used by end-users of the SciDB system today and, therefore, getting performance data around the CSV to SciDB load process was desirable. Further work might include diving deeper into alternative options, including the possibility of using a binary load facility, may help decrease the time needed to get data into SciDB for analysis. Still, the overall load times using the method outlined in this project is already competitive with commercial data warehouse products on the market today.

It appears as though all benchmarks start to converge to some lower bound for processing time that adding more instances will not significantly decrease. Further profiling the system under those circumstances may lead to insights into increasing overall system performance. Chunk sizing, chunk cell density and cluster size can interact in ways that are difficult to predict since the interactions may be data-dependent or affected by the nature of the ad-hoc query being run. Additional study may yield some generalizations that can help prevent performance fluctuations and performance regressions. From an Earth Science perspective, some additional

functionality can be added to SciDB, such as support for overlapping window aggregates and geographical projection library support, which will further attract members of the community to the database and also further increase performance of the system using optimized, well-tested components.

While the benchmarks for this project were limited to well-known, relatively simple analytical tasks, they represent the most typical operations performed in the domain: filtering data by time and location, time-series aggregations, comparing and performing arithmetic across multiple bands of data, and so on. Once data has been loaded into MODBASE, the analysis that can be performed is practically unbounded. The SciDB system comes loaded with an ever-growing library of mathematical functions and supports user-defined functions and data types. Earth scientists wishing to perform analysis of MODIS data now have a powerful tool in their belt that can import MODIS Level 1B data and perform analysis that can diverge from the NASA packaged product set.

Appendix: Benchmark Server Specifications

This section details the configuration for the server used to perform all performance benchmarks. The server was dedicated to the task of running benchmarks and had no virtualization, additional users, or extraneous processes running beyond the core operating system processes.

Model
Dell PowerEdge R710 – 2U

Operating System
Ubuntu Server 10.10

SciDB Version
12.7

Processors
(2) Intel Xeon E5645 2.40 GHz, 12M cache, 5.86 GT/S QPI, 6 hyper-threaded cores each. 24 total threads.

RAM
48GB (6 x 8GB), 1333MHz, Dual Ranked LV RDIMMS

Hard Drives
(6) 2TB 7.2K RPM, Near-Line SAS 6Gbps 3.5in Hot-plug Hard Drive

RAID
PERC H700 Integrated RAID Controller, 512MB Cache, x6
(3) Raid 0 volumes were configured using the 3 pairs of physical drives and 1 MB stripe size.

Ethernet
Embedded Broadcom, GB Ethernet NICS with TOE

Power Supply
High Output Power Supply Redundant, 870W (330-3475)

Appendix: SciDB Configuration File

This section provides the contents of the *config.ini* file used to configure SciDB for benchmarking purposes. The only difference between the configurations for single and multiple instance benchmarking runs is the number after the comma on the first line, affecting the number of SciDB instances that are initialized.

```
[modisdb]
server-0=localhost,0
db_user=singleinstance
db_passwd=singleinstance
install_root=/opt/scidb/12.7
metadata=/opt/scidb/12.7/share/scidb/meta.sql
pluginsdir=/opt/scidb/12.7/lib/scidb/plugins
logconf=/opt/scidb/12.7/share/scidb/log4cxx.properties
base-path=/data/scidb
base-port=1239
interface=eth0
chunk-segment-size=250000000
io-log-threshold=-1
result-prefetch-threads=4
execution-threads=8
result-prefetch-queue-size=4
chunk-reserve=0
tmp-path=/data/scidb/tmp
merge-sort-buffer=1024
smgr-cache-size=1024
mem-array-threshold=1024
rle-chunk-format=true
repart-use-sparse-algorithm=true
```

Appendix: CSV Sample Data

This section provides excerpts of CSV that has been output from the mod2csv utility. Note that only Band_1_Measurements.csv is shown as it has the same structure as all Band_x_Measurements files.

Granule_Metadata.csv

```
start_time,platform_id,resolution_id,scans,track_measurements,scan_measurements,day_night_flag,fi
le_id,geo_file_id
201201010000,1,2,203,2030,1354,"Day","MOD021KM.A2012001.0000.005.2012001080421.hdf","MOD03.A20120
01.0000.005.2012001071428.hdf"
```

Band_Metadata.csv

```
start_time,platform_id,resolution_id,band_id,radiance_scale,radiance_offset,reflectance_scale,ref
lectance_offset,corrected_counts_scale,corrected_counts_offset,specified_uncertainty,uncertainty_
scaling_factor
201201010000,1,2,0,0.026254319,-0.0,4.965319E-5,-0.0,0.1249733,-0.0,1.5,7.0
201201010000,1,2,1,0.009791818,-0.0,2.9977824E-5,-0.0,0.1249733,-0.0,1.5,7.0
201201010000,1,2,2,0.034223136,-0.0,4.978973E-5,-0.0,0.1249733,-0.0,1.5,7.0
201201010000,1,2,3,0.023708515,-0.0,3.859623E-5,-0.0,0.1249733,-0.0,1.5,7.0
201201010000,1,2,4,0.005847191,-0.0,3.744481E-5,-0.0,0.1249733,-0.0,1.5,5.0
201201010000,1,2,5,0.0026472483,-0.0,3.3473836E-5,-0.0,0.1249733,-0.0,1.5,5.0
201201010000,1,2,6,8.1179396E-4,-0.0,2.7300814E-5,-0.0,0.1249733,-0.0,1.5,5.0
201201010000,1,2,7,0.013242942,316.9722,2.3043067E-5,316.9722,0.12619403,316.9722,1.5,7.0
201201010000,1,2,8,0.0084292535,316.9722,1.3449697E-5,316.9722,0.12619403,316.9722,1.5,7.0
201201010000,1,2,9,0.005361448,316.9722,8.221454E-6,316.9722,0.12619403,316.9722,1.5,7.0
...
```

Geodata.csv

```
longitude_e4,latitude_e4,start_time,platform_id,resolution_id,track_index,scan_index,height,senso
r_zenith,sensor_azimuth,range,solar_zenith,solar_azimuth,land_sea_mask
1504773,390042,201201010000,1,2,0,0,19,65.53,92.72,1419750,68.23,150.02,7
1505331,390023,201201010000,1,2,0,1,19,65.41,92.75,1415325,68.21,150.07,7
1505885,390004,201201010000,1,2,0,2,19,65.28,92.79,1410975,68.19,150.12,7
1506435,389984,201201010000,1,2,0,3,19,65.16,92.82,1406650,68.16,150.17,7
1506980,389964,201201010000,1,2,0,4,19,65.04,92.86,1402350,68.14,150.22,7
1507521,389944,201201010000,1,2,0,5,19,64.91,92.89,1398100,68.12,150.27,7
1508058,389924,201201010000,1,2,0,6,19,64.79,92.93,1393875,68.09,150.32,7
1508591,389904,201201010000,1,2,0,7,19,64.67,92.96,1389700,68.07,150.36,7
1509120,389884,201201010000,1,2,0,8,19,64.54,93,1385550,68.05,150.41,7
...
```

Band_1_Measurements.csv

```
longitude_e4,latitude_e4,start_time,platform_id,resolution_id,band_id,si_value,radiance,reflectan
ce,uncertainty_index,uncertainty_pct
1504773,390042,201201010000,1,2,0,3614,94.88311,0.17944662,2,1.996
1505331,390023,201201010000,1,2,0,3668,96.30084,0.18212791,2,1.996
1505885,390004,201201010000,1,2,0,4282,112.42099,0.21261495,1,1.73
1506435,389984,201201010000,1,2,0,2783,73.065765,0.13818483,3,2.303
1506980,389964,201201010000,1,2,0,1467,38.515087,0.07284123,5,3.064
1507521,389944,201201010000,1,2,0,1016,26.674387,0.050447643,6,3.535
1508058,389924,201201010000,1,2,0,988,25.939266,0.049057353,6,3.535
1508591,389904,201201010000,1,2,0,971,25.492943,0.048213247,6,3.535
1509120,389884,201201010000,1,2,0,1452,38.12127,0.07209643,5,3.064
...
```

Appendix: SciDB One-Dimensional Load Schema

This section provides the definitions for each temporary load array used in this project. These arrays temporarily store the one-dimensional loaded CSV data. Each load array can then be re-dimensioned in to its corresponding multidimensional analysis array. The band_x_measurements array actually represents 38 identically defined arrays where x is replaced with the MODIS name for the band (e.g., 1, 13lo, 32). Each band_x_measurement array will contain measurements only for its corresponding band. The decision to separate band data in such a manner was an intentional, selective denormalization that allows greater parallelization.

```
CREATE IMMUTABLE EMPTY ARRAY load_granule_metadata
<
start_time : int64,
platform_id : int64,
resolution_id : int64,
scans : uint8,
track_measurements : uint16,
scan_measurements : uint16,
day_night_flag : string,
file_id : string,
geo_file_id : string
>
[
i = 0 : *, 500000, 0
];
```

```
CREATE IMMUTABLE EMPTY ARRAY load_band_metadata
<
start_time : int64,
platform_id : int64,
resolution_id : int64,
band_id : int64,
radiance_scale : double,
radiance_offset : float,
reflectance_scale : double,
reflectance_offset : float,
corrected_counts_scale : double,
corrected_counts_offset : float,
specified_uncertainty : float,
uncertainty_scaling_factor : float
>
[
i = 0 : *, 500000, 0
];
```

SciDB Load Schema [continued]

```
CREATE IMMUTABLE EMPTY ARRAY load_geodata
<
longitude_e4 : int64,
latitude_e4 : int64,
start_time : int64,
platform_id : int64,
resolution_id : int64,
track_index : int16,
scan_index : int16,
height : int16,
sensor_zenith : float,
sensor_azimuth : float,
range : uint32,
solar_zenith : float,
solar_azimuth : float,
land_sea_mask : uint8
>
[
i = 0 : *, 500000, 0
];
```

```
CREATE IMMUTABLE EMPTY ARRAY load_band_x_measurements
<
longitude_e4 : int64,
latitude_e4 : int64,
start_time : int64,
platform_id : int64,
resolution_id : int64,
si_value : uint16,
radiance : double,
reflectance : double,
uncertainty_index : uint8,
uncertainty_pct : float
>
[
i = 0 : *, 500000, 0
];
```

Appendix: SciDB Multidimensional Analysis Schema

This section provides the definitions for each array used in this project. The platforms, resolutions and bands arrays are only reference arrays that give meaningful descriptions for all id values used in the other arrays. All arrays are marked immutable as they are designed so that the values in a given chunk never change once imported. This helps to minimize overhead and accidental updates to data. The band_x_measurements array represents 38 identically defined arrays where x is replaced with the MODIS name for the band (e.g., 1, 13lo, 32). Each band_x_measurement array will contain measurements only for its corresponding band. The decision to separate band data in such a manner was an intentional, selective denormalization that allows greater parallelization.

```
CREATE IMMUTABLE EMPTY ARRAY platforms
<
description : string
>
[
platform_id = 0 : 1, 1, 0
];


CREATE IMMUTABLE EMPTY ARRAY resolutions
<
description : string
>
[
resolution_id = 0 : 2, 1, 0
];


CREATE IMMUTABLE EMPTY ARRAY bands
<
description : string
>
[
band_id = 0 : 37, 38, 0
];


CREATE IMMUTABLE EMPTY ARRAY granule_metadata
<
scans : uint8,
track_measurements : uint16,
scan_measurements : uint16,
day_night_flag : string,
file_id : string,
geo_file_id : string
>
[
start_time = 199900000000 : 201400000000, 1, 0,
platform_id = 0 : 1, 1, 0,
resolution_id = 0 : 2, 1, 0
];
```

SciDB Multidimensional Analysis Schema [continued]

```
CREATE IMMUTABLE EMPTY ARRAY band_metadata
<
radiance_scale : double,
radiance_offset : float,
reflectance_scale : double,
reflectance_offset : float,
corrected_counts_scale : double,
corrected_counts_offset : float,
specified_uncertainty : float,
uncertainty_scaling_factor : float
>
[
start_time = 199900000000 : 201400000000, 1, 0,
platform_id = 0 : 1, 1, 0,
resolution_id = 0 : 2, 1, 0,
band_id = 0 : 37, 38, 0
];


CREATE IMMUTABLE EMPTY ARRAY geodata
<
track_index : int16,
scan_index : int16,
height : int16,
sensor_zenith : float,
sensor_azimuth : float,
range : uint32,
solar_zenith : float,
solar_azimuth : float,
land_sea_mask : uint8
>
[
longitude_e4 = -1800000 : 1800000, 50000, 0,
latitude_e4 = -900000 : 900000, 50000, 0,
start_time = 199900000000 : 201400000000, 1, 0,
platform_id = 0 : 1, 1, 0,
resolution_id = 0 : 2, 1, 0
];


CREATE IMMUTABLE EMPTY ARRAY band_x_measurements
<
si_value : uint16,
radiance : double,
reflectance : double,
uncertainty_index : uint8,
uncertainty_pct : float
>
[
longitude_e4 = -1800000 : 1800000, 50000, 0,
latitude_e4 = -900000 : 900000, 50000, 0,
start_time = 199900000000 : 201400000000, 1, 0,
platform_id = 0 : 1, 1, 0,
resolution_id = 0 : 2, 1, 0
];
```

Appendix: SciDB Reference Array Data

This section provides the data that is loaded into the 3 reference arrays in the SciDB analysis schema. The values are shown in SciDB dense load format. Once loaded, these values never change in SciDB. The corresponding ID values are assigned based on the order of the values, starting with 0 (e.g., platform_id: 0 <=> "Aqua", resolution_id: 0 <=> "250m").

Bands

```
[
("1"),("2"),("3"),("4"),("5"),("6"),("7"),("8"),("9"),("10"),("11"),("12"),
("13lo"),("13hi"),("14lo"),("14hi"),("15"),("16"),("17"),("18"),("19"),
("20"),("21"),("22"),("23"),("24"),("25"),("26"),("27"),("28"),("29"),("30"),("31"),
("32"),("33"),("34"),("35"),("36")
]
```

Platforms

```
[ ("Aqua") ];
[ ("Terra") ]
```

Resolutions

```
[ ("250m") ];
[ ("500m") ];
[ ("1km") ]
```

Appendix: List of MODIS Granules Used

Below is a list of all NASA MODIS HDF granules that were used in this project.

1km Granules

163MB - MOD021KM.A2012092.1925.005.2012093150459.hdf
161MB - MOD021KM.A2012093.1830.005.2012094181629.hdf
164MB - MOD021KM.A2012094.1910.005.2012095025321.hdf
167MB - MOD021KM.A2012094.1915.005.2012095025322.hdf
162MB - MOD021KM.A2012095.1815.005.2012096183440.hdf
164MB - MOD021KM.A2012095.1820.005.2012096183319.hdf
160MB - MOD021KM.A2012097.1805.005.2012100085003.hdf
158MB - MOD021KM.A2012097.1945.005.2012100084601.hdf
161MB - MOD021KM.A2012098.1850.005.2012100095757.hdf
161MB - MOD021KM.A2012099.1755.005.2012100104612.hdf
164MB - MOD021KM.A2012099.1930.005.2012100104839.hdf
162MB - MOD021KM.A2012100.1835.005.2012101024852.hdf
169MB - MOD021KM.A2012101.1920.005.2012102015232.hdf
165MB - MOD021KM.A2012102.1825.005.2012103015211.hdf
164MB - MOD021KM.A2012103.1905.005.2012104014458.hdf
161MB - MOD021KM.A2012104.1810.005.2012107084207.hdf
163MB - MOD021KM.A2012104.1950.005.2012107083422.hdf
163MB - MOD021KM.A2012105.1855.005.2012107095315.hdf
161MB - MOD021KM.A2012106.1800.005.2012107103510.hdf
153MB - MOD021KM.A2012106.1935.005.2012107103916.hdf
158MB - MOD021KM.A2012106.1940.005.2012107103749.hdf
157MB - MOD021KM.A2012107.1840.005.2012108023204.hdf
154MB - MOD021KM.A2012107.1845.005.2012108023020.hdf

500m Granules

161MB - MOD02HKM.A2012092.1925.005.2012093150459.hdf
156MB - MOD02HKM.A2012093.1830.005.2012094181629.hdf
163MB - MOD02HKM.A2012094.1910.005.2012095025321.hdf
157MB - MOD02HKM.A2012094.1915.005.2012095025322.hdf
159MB - MOD02HKM.A2012095.1815.005.2012096183440.hdf
157MB - MOD02HKM.A2012095.1820.005.2012096183319.hdf
159MB - MOD02HKM.A2012097.1805.005.2012100085003.hdf
162MB - MOD02HKM.A2012097.1945.005.2012100084601.hdf
154MB - MOD02HKM.A2012098.1850.005.2012100095757.hdf
156MB - MOD02HKM.A2012099.1755.005.2012100104612.hdf
162MB - MOD02HKM.A2012099.1930.005.2012100104839.hdf
157MB - MOD02HKM.A2012100.1835.005.2012101024852.hdf
160MB - MOD02HKM.A2012101.1920.005.2012102015232.hdf
158MB - MOD02HKM.A2012102.1825.005.2012103015211.hdf
169MB - MOD02HKM.A2012103.1905.005.2012104014458.hdf
163MB - MOD02HKM.A2012104.1810.005.2012107084207.hdf
159MB - MOD02HKM.A2012104.1950.005.2012107083422.hdf
166MB - MOD02HKM.A2012105.1855.005.2012107095315.hdf
161MB - MOD02HKM.A2012106.1800.005.2012107103510.hdf
165MB - MOD02HKM.A2012106.1935.005.2012107103916.hdf
164MB - MOD02HKM.A2012106.1940.005.2012107103749.hdf
164MB - MOD02HKM.A2012107.1840.005.2012108023204.hdf
158MB - MOD02HKM.A2012107.1845.005.2012108023020.hdf

250m Granules

177MB - MOD02QKM.A2012092.1925.005.2012093150459.hdf
171MB - MOD02QKM.A2012093.1830.005.2012094181629.hdf
180MB - MOD02QKM.A2012094.1910.005.2012095025321.hdf
171MB - MOD02QKM.A2012094.1915.005.2012095025322.hdf
176MB - MOD02QKM.A2012095.1815.005.2012096183440.hdf
173MB - MOD02QKM.A2012095.1820.005.2012096183319.hdf
175MB - MOD02QKM.A2012097.1805.005.2012100085003.hdf
178MB - MOD02QKM.A2012097.1945.005.2012100084601.hdf
167MB - MOD02QKM.A2012098.1850.005.2012100095757.hdf
170MB - MOD02QKM.A2012099.1755.005.2012100104612.hdf
179MB - MOD02QKM.A2012099.1930.005.2012100104839.hdf
173MB - MOD02QKM.A2012100.1835.005.2012101024852.hdf
176MB - MOD02QKM.A2012101.1920.005.2012102015232.hdf
171MB - MOD02QKM.A2012102.1825.005.2012103015211.hdf
187MB - MOD02QKM.A2012103.1905.005.2012104014458.hdf
179MB - MOD02QKM.A2012104.1810.005.2012107084207.hdf
175MB - MOD02QKM.A2012104.1950.005.2012107083422.hdf
183MB - MOD02QKM.A2012105.1855.005.2012107095315.hdf
177MB - MOD02QKM.A2012106.1800.005.2012107103510.hdf
180MB - MOD02QKM.A2012106.1935.005.2012107103916.hdf
181MB - MOD02QKM.A2012106.1940.005.2012107103749.hdf
181MB - MOD02QKM.A2012107.1840.005.2012108023204.hdf
174MB - MOD02QKM.A2012107.1845.005.2012108023020.hdf

Geolocation Granules

29MB - MOD03.A2012092.1925.005.2012093141114.hdf
32MB - MOD03.A2012093.1830.005.2012094172529.hdf
32MB - MOD03.A2012094.1910.005.2012095023114.hdf
27MB - MOD03.A2012094.1915.005.2012095023107.hdf
32MB - MOD03.A2012095.1815.005.2012096174413.hdf
29MB - MOD03.A2012095.1820.005.2012096174418.hdf
32MB - MOD03.A2012097.1805.005.2012098011338.hdf
27MB - MOD03.A2012097.1945.005.2012098011322.hdf
29MB - MOD03.A2012098.1850.005.2012099012734.hdf
31MB - MOD03.A2012099.1755.005.2012100005923.hdf
30MB - MOD03.A2012099.1930.005.2012100005914.hdf
32MB - MOD03.A2012100.1835.005.2012101020833.hdf
28MB - MOD03.A2012101.1920.005.2012102013603.hdf
31MB - MOD03.A2012102.1825.005.2012103014540.hdf
32MB - MOD03.A2012103.1905.005.2012104012659.hdf
33MB - MOD03.A2012104.1810.005.2012105012255.hdf
27MB - MOD03.A2012104.1950.005.2012105012234.hdf
30MB - MOD03.A2012105.1855.005.2012106011503.hdf
32MB - MOD03.A2012106.1800.005.2012107011640.hdf
31MB - MOD03.A2012106.1935.005.2012107011555.hdf
27MB - MOD03.A2012106.1940.005.2012107011609.hdf
33MB - MOD03.A2012107.1840.005.2012108020205.hdf
28MB - MOD03.A2012107.1845.005.2012108020153.hdf

Appendix: RGB Composite Benchmark Source

Below is the shell script used to process 30° x 30° RGB composite data.

```
#!/bin/bash
platform=1
resolution=1
startLongitude_e4=-1280000
stopLongitude_e4=-980001
startLatitude_e4=150000
stopLatitude_e4=449999
startTime=201204111825
stopTime=201204111825

iquery -anq "remove(rgb)" > /dev/null 2>&1
attrs="<red:double null,green:double null,blue:double null>"
dims="[x=1:300000,15000,0,y=1:300000,15000,0]"
schema=$attrs$dims
iquery -anq "create immutable empty array rgb $schema" > /dev/null 2>&1

echo "Building RGB composite values..."
time iquery -anq "
  redimension_store(
    apply(
      between(
        join(
          attribute_rename(
            band_1_measurements,
            reflectance,
            red
          ),
          join(
            attribute_rename(
              band_4_measurements,
              reflectance,
              green
            ),
            attribute_rename(
              band_3_measurements,
              reflectance,
              blue
            )
          )
        ),
        $startLongitude_e4, $startLatitude_e4, $startTime, $platform, $resolution,
        $stopLongitude_e4, $stopLatitude_e4, $stopTime, $platform, $resolution
      ),
      x, longitude_e4 - $startLongitude_e4 + 1,
      y, latitude_e4 - $startLatitude_e4 + 1
    ),
    rgb,
    avg(red) as red,
    avg(green) as green,
    avg(blue) as blue
  )"
```

Appendix: NDVI Benchmark Source

Below is the shell script used to process 30° x 30° NDVI data.

```
#!/bin/bash
platform=1
resolution=1
startLongitude_e4=-1280000
stopLongitude_e4=-980001
startLatitude_e4=150000
stopLatitude_e4=449999
startTime=201200000000
stopTime=201300000000

iquery -anq "remove(ndvi)" > /dev/null 2>&1
attrs="<ndvi:double null>"

dims="[x=1:300000,10000,0,y=1:300000,10000,0]"
schema=$attrs$dims
iquery -anq "create immutable empty array ndvi $schema" > /dev/null 2>&1

echo "Calculating NDVI values..."
time iquery -anq "
  redimension_store(
    apply(
      between(
        join(
          attribute_rename(
            band_1_measurements,
            reflectance,
            red
          ),
          attribute_rename(
            band_2_measurements,
            reflectance,
            nir
          )
        ),
        $startLongitude_e4, $startLatitude_e4, $startTime, $platform, $resolution,
        $stopLongitude_e4, $stopLatitude_e4, $stopTime, $platform, $resolution
      ),
      x, longitude_e4 - $startLongitude_e4 + 1,
      y, latitude_e4 - $startLatitude_e4 + 1,
      ndvi, (nir - red) / (nir + red)
    ),
    ndvi,
    max(ndvi) as ndvi
  )"
```

Appendix: NDSI+Snowmap Benchmark Source

Below is the shell script used to process 30° x 30° NDSI+Snowmap data.

```
#!/bin/bash
platform=1
resolution=1
startLongitude_e4=-1280000
stopLongitude_e4=-980001
startLatitude_e4=150000
stopLatitude_e4=449999
startTime=201200000000
stopTime=201300000000

iquery -anq "remove(ndsi)" > /dev/null 2>&1
attrs="<ndsi:double null,snowfraction:double null>"
dims="[x=1:300000,10000,0,y=1:300000,10000,0]"
schema=$attrs$dims
iquery -anq "create immutable empty array ndsi $schema" > /dev/null 2>&1

echo "Calculating NDSI values..."
time iquery -anq "
  redimension_store(
    apply(
      apply(
        between(
          join(
            attribute_rename(
              band_2_measurements,
              reflectance,
              band2
            ),
            join(
              attribute_rename(
                band_4_measurements,
                reflectance,
                band4
              ),
              attribute_rename(
                band_6_measurements,
                reflectance,
                band6
              )
            )
          ),
          $startLongitude_e4, $startLatitude_e4, $startTime, $platform, $resolution,
          $stopLongitude_e4, $stopLatitude_e4, $stopTime, $platform, $resolution
        ),
        x, longitude_e4 - $startLongitude_e4 + 1,
        y, latitude_e4 - $startLatitude_e4 + 1,
        ndsi, (band4 - band6) / (band4 + band6)
      ),
      snowmap, iif(ndsi > 0.4 and band2 > 0.11 and band4 > 0.1, 1.0, 0.0)
    ),
    ndsi,
    max(ndsi) as ndsi,
    max(snowmap) as snowmap
  )"
```

Appendix: Gridding Benchmark Source

Below is the script used to grid the 30° x 30° NDVI data (part of the NDVI benchmark script).

```
echo "Generating NDVI grid centers..."
xTiles=1500
xCellsPerTile=200
yTiles=1500
yCellsPerTile=200

overlapCells=50
xBlurWindowSize=$(($xCellsPerTile + $overlapCells))
yBlurWindowSize=$(($yCellsPerTile + $overlapCells))

iquery -anq "remove(grid_centers)" > /dev/null 2>&1
iquery -anq "create immutable empty array grid_centers $schema" > /dev/null 2>&1

time iquery -anq "
  redimension_store(
    apply(
      cross(
        build(<x:int64>[i=1:$xTiles,$xTiles,0],
              int64($xCellsPerTile * (i - 0.5) + 0.5)),
        build(<y:int64>[i=1:$yTiles,$yTiles,0],
              int64($yCellsPerTile * (i - 0.5) + 0.5))
      ),
      ndvi, double(null)
    ),
    grid_centers
  )"

echo
echo "Gridding NDVI results..."
iquery -anq "remove(ndvi_gridded)" > /dev/null 2>&1
xStart=$(echo $xCellsPerTile 2 | awk '{printf "%d", int($1 / $2 + 0.5)}')
yStart=$(echo $yCellsPerTile 2 | awk '{printf "%d", int($1 / $2 + 0.5)}')

time iquery -anq "
  store(
    thin(
      window(
        merge(ndvi, grid_centers),
        $xBlurWindowSize,
        $yBlurWindowSize,
        avg(ndvi) as ndvi
      ),
      $xStart, $xCellsPerTile,
      $yStart, $yCellsPerTile
    ),
    ndvi_gridded
  )"

echo
echo "Exporting results to CSV..."
time iquery -aq "scan(ndvi_gridded)" > ~/ndvi_gridded.csv

iquery -anq "remove(grid_centers)" > /dev/null 2>&1
```

Appendix: Raw Benchmark Data

Instances	Window Size	Run	RGB	NDVI	NDSI	Gridding
1	10° x 10°	1	33.1	158.5	305.3	533.7
1	10° x 10°	2	29.8	141.5	272.4	518.7
1	10° x 10°	3	29.7	141.7	287.8	466.3
1	**10° x 10°**	**Avg**	**30.9**	**147.2**	**288.5**	**506.2**
1	20° x 20°	1	71.4	494.4	967.7	1541.9
1	20° x 20°	2	67.8	447.3	946.8	1547.9
1	20° x 20°	3	65.2	447.0	912.3	1579.3
1	**20° x 20°**	**Avg**	**68.1**	**462.9**	**942.3**	**1556.4**
1	30° x 30°	1	81.9	815.4	1691.8	2443.7
1	30° x 30°	2	77.5	769.3	1627.2	2703.7
1	30° x 30°	3	80.2	768.9	1506.4	2753.2
1	**30° x 30°**	**Avg**	**79.9**	**784.5**	**1608.5**	**2633.5**
2	10° x 10°	1	18.5	92.8	168.7	324.2
2	10° x 10°	2	26.0	84.9	160.0	307.5
2	10° x 10°	3	15.8	84.0	165.1	324.9
2	**10° x 10°**	**Avg**	**20.1**	**87.2**	**164.6**	**318.9**
2	20° x 20°	1	38.1	306.2	556.9	1163.8
2	20° x 20°	2	35.1	271.2	535.8	1145.1
2	20° x 20°	3	36.2	272.2	516.4	1034.3
2	**20° x 20°**	**Avg**	**36.5**	**283.2**	**536.4**	**1114.4**
2	30° x 30°	1	43.2	492.9	953.7	1847.5
2	30° x 30°	2	41.8	471.2	912.7	1906.0
2	30° x 30°	3	41.8	466.9	951.4	1875.5
2	**30° x 30°**	**Avg**	**42.3**	**477.0**	**939.3**	**1876.3**
3	10° x 10°	1	17.9	86.4	186.5	215.2
3	10° x 10°	2	14.7	79.1	140.4	194.9
3	10° x 10°	3	16.7	74.4	140.6	215.0
3	**10° x 10°**	**Avg**	**16.4**	**80.0**	**155.8**	**208.4**
3	20° x 20°	1	35.8	291.0	445.4	781.5
3	20° x 20°	2	28.1	226.2	422.0	696.4
3	20° x 20°	3	27.3	247.4	432.0	752.6
3	**20° x 20°**	**Avg**	**30.4**	**254.9**	**433.1**	**743.5**
3	30° x 30°	1	35.8	410.0	752.2	1234.9
3	30° x 30°	2	34.4	400.7	743.1	1305.4
3	30° x 30°	3	31.5	388.1	719.4	1269.0
3	**30° x 30°**	**Avg**	**33.9**	**399.6**	**738.2**	**1269.8**
4	10° x 10°	1	15.9	84.6	127.8	186.1
4	10° x 10°	2	10.5	61.7	98.8	190.2
4	10° x 10°	3	10.1	58.4	100.5	190.9
4	**10° x 10°**	**Avg**	**12.2**	**68.2**	**109.0**	**189.1**
4	20° x 20°	1	28.8	233.3	403.7	622.2
4	20° x 20°	2	21.1	185.6	321.8	648.0
4	20° x 20°	3	28.6	190.0	323.5	610.4
4	**20° x 20°**	**Avg**	**26.2**	**203.0**	**349.7**	**626.9**
4	30° x 30°	1	36.0	375.0	661.6	1082.0
4	30° x 30°	2	24.2	386.8	626.7	1042.8
4	30° x 30°	3	24.5	383.9	583.8	1031.0
4	**30° x 30°**	**Avg**	**28.2**	**381.9**	**624.0**	**1051.9**
5	10° x 10°	1	24.1	85.1	134.6	170.9
5	10° x 10°	2	22.0	56.8	104.3	170.4
5	10° x 10°	3	22.0	56.6	102.7	167.5
5	**10° x 10°**	**Avg**	**22.7**	**66.2**	**113.9**	**169.6**
5	20° x 20°	1	38.8	233.4	320.1	589.8
5	20° x 20°	2	32.7	175.7	282.6	568.3
5	20° x 20°	3	31.3	159.7	316.8	573.5
5	**20° x 20°**	**Avg**	**34.3**	**189.6**	**306.5**	**577.2**
5	30° x 30°	1	36.7	329.7	547.9	965.5
5	30° x 30°	2	36.7	366.2	589.0	1022.7
5	30° x 30°	3	36.9	347.3	578.1	957.6
5	**30° x 30°**	**Avg**	**36.8**	**347.7**	**571.7**	**981.9**

Appendix: MATLAB Visualization Source

Display RGB Composite Image in MATLAB

```
raw=importdata('~/gridded_data.csv', ',', 1);
sorted=sortrows(raw.data);
red_sc=scaledata(sorted(:,3), 0, 1);
red_img=reshape(red_sc, 2000, 2000);
green_sc=scaledata(sorted (:,4), 0, 1);
green_img=reshape(green_sc, 2000, 2000);
blue_sc=scaledata(sorted (:,5), 0, 1);
blue_img=reshape(blue_sc, 2000, 2000);
image(cat(3, red_img, green_img, blue_img));
axis image;
axis xy;
axis off;
```

Display NDVI Image in MATLAB

```
raw=importdata('~/gridded_data.csv', ',', 1);
sorted=sortrows(raw.data);
img=reshape(sorted(:,3), 500, 500);
imagesc(img);
axis image;
axis xy;
axis off;
```

Display NDSI or Snow Map Image in MATLAB

```
raw=importdata('~/gridded_data.csv', ',', 1);
sorted=sortrows(raw.data);
img=reshape(sorted(:,3), 500, 500);
imagesc(img);
axis image;
axis xy;
axis off;

or (for snowmap):

img=reshape(sorted(:,4), 500, 500);
imagesc(img);
axis image;
axis xy;
axis off;
```

Bibliography

[1] Adam Jacobs, "The Pathologies of Big Data," *ACM Queue*, vol. 7, no. 6, pp. 36-44, July 2009.

[2] Michael Stonebraker and Ugur Cetintemel, "One Size Fits All: An Idea whose Time has Come and Gone," in *Proceedings of the 21st International Conference on Data Engineering (ICDE 2005)*, 2005, pp. 2-11.

[3] Michael Stonebraker, Samuel Madden, Daniel J. Abadi, Nabil Hachem, and Pat Helland, "The End of an Architectural Era (It's Time for a Complete Rewrite)," in *Proceedings of the 33rd International Conference on Very Large Databases (VLDB 2007)*, 2007, pp. 1150-1160.

[4] Andrew Pavlo et al., "A Comparison of Approaches to Large-Scale Data Analysis," in *Proceedings of the 35th SIGMOD International Conference on Management of Data (SIGMOD '09)*, 2009, pp. 165-178.

[5] Michael Stonebraker et al., "MapReduce and Parallel DBMSs: Friends or Foes?," *Communications of the ACM*, vol. 53, no. 1, pp. 64-71, January 2010.

[6] XLDB. About the XLDB Organization. [Online]. http://www.xldb.org/about/

[7] Michael Way. (2011, August) Challenges for LSST scale data sets. [Online]. http://arxiv.org/abs/1108.5124v1

[8] Michael Stonebraker et al., "Requirements for Science Data Bases and SciDB," in *Proceedings of the 2009 Conference on Innovative Data Systems Research (CIDR 2009)*, 2009.

[9] Philippe Cudre-Mauroux et al. (2009) SS-DB: A Standard Science DBMS Benchmark.

[10] Philippe Cudre-Mauroux et al., "A Demonstration of SciDB: A Science-Oriented DBMS," in *Proceedings of the 2009 VLDB Endowment (August 2009)*, Lyon, France, 2009, pp. 1534-1537.

[11] Paul Brown et al., "Overview of SciDB: Large Scale Array Storage, Processing and Analysis," in *Proceedings of the 2010 International Conference on Management of Data (SIGMOD 2010)*, 2010, pp. 963-968.

[12] Michael Stonebraker, Paul Brown, Alex Poliakov, and Suchi Raman, "The Architecture of SciDB," in *Proceedings of the 23rd International Conference on Scientific and Statistical Database Management (SSDBM 2011)*, 2011, pp. 1-16.

[13] SciDB. (2012) SciDB User's Guide.

[14] NASA. Ocean Motion. [Online]. http://oceanmotion.org/guides/fd_3/fd_student_3.htm

[15] NASA. MODIS. [Online]. http://modis.gsfc.nasa.gov/

[16] NASA. EOS Program. [Online]. http://eospso.gsfc.nasa.gov/

[17] NASA. Terra Satellite. [Online]. http://terra.nasa.gov/

[18] NASA. Aqua Satellite. [Online]. http://aqua.nasa.gov/

[19] NASA. MODIS Data Products. [Online]. http://modis.gsfc.nasa.gov/data/dataprod/index.php

[20] NASA. Land Processes Active Distributed Archive Center. [Online]. https://lpdaac.usgs.gov/tools

[21] NASA. MODIS Brochure. [Online]. http://modis.gsfc.nasa.gov/about/media/modis_brochure.pdf

[22] The HDF Group. HDF-EOS Tools and Information Center. [Online]. http://hdfeos.org/

[23] The HDF Group. [Online]. http://www.hdfgroup.org/

[24] HDF Group. HDF 5 Tools and Software. [Online]. http://www.hdfgroup.org/products/hdf5_tools/index.html

[25] Michael Stonebraker, "An Overview of the Sequoia 2000 Project," in *Compcon Spring '92. 37th IEEE Computer Society International Conference, Digest of Papers.*, 1992, pp. 383-388.

[26] Michael Stonebraker, Jim Frew, Kenn Gardels, and Jeff Meredith, "The SEQUOIA 2000 Storage Benchmark," in *SIGMOD '93 Proceedings of teh 1993 ACM SIGMOD International Conference on Management of Data*, 1993, pp. 2-11.

[27] Jack Becla, Daniel Wang, and Kian-Tat Lim. (2012, February) SLAC National Accelerator Laboratory. [Online]. http://slac.stanford.edu/pubs/slactns/tn05/slac-tn-11-020.pdf

[28] SciDB. SciDB Use Cases. [Online]. http://www.scidb.org/use/

[29] Wenli Yang and Liping Di, "An Accurate and Automated Approach to Georectification of HDF-EOS Swath Data," *Photogrammetric Engineering and Remote Sensing*, vol. 70, no. 4, pp. 397-404, April 2004.

[30] NASA. (1997, August) MODIS Level 1A Earth Location: Algorithm Theoretical Basis Document Version 3.0. [Online]. http://modis.gsfc.nasa.gov/data/atbd/atbd_mod28_v3.pdf

[31] Nicolas Pascal. (2009, April) iCare Wiki, Universite Lille1. [Online]. http://www.icare.univ-lille1.fr/wiki/index.php/MODIS_geolocation

[32] Consultative Committee for Space Data Systems. (2000, June) Parameter Value Language Specification. [Online]. http://public.ccsds.org/publications/archive/641x0b2.pdf

[33] NASA. (2006, August) MODIS Level 1B Product User's Guide. [Online]. http://mcst.gsfc.nasa.gov/sites/mcst.gsfc/files/file_attachments/M1054.pdf

[34] Daniel Liwei Want and Jacek Becla. A Basic Loader of HDF Files into SciDB. [Online]. https://github.com/wangd/SciDB-HDF5

[35] Dorothy K. Hall, George A. Riggs, and Vincent V. Salomonson. (2001, September) Algorithm Theoretical Basis Document (ATBD) for the MODIS Snow and Sea Ice-Mapping Algorithms. [Online]. http://modis.gsfc.nasa.gov/data/atbd/atbd_mod10.pdf